WINDOWS COMMUNICATION FOUNDATION
SVILUPPA IL TUO PRIMO "SERVIZIO" BASATO SUL NUOVO FRAMEWORK DI MICROSOFT. UN'APPLICAZIONE PER LA GESTIONE DI UN BOOKSTORE
IN PIÙ LA VIDEOGUIDA NEL CD

Rivista + "Le grandi guide di ioProgrammo" N°6 a € 12,90 in più
VERSIONE PLUS RIVISTA+LIBRO+CD €9,90
VERSIONE STANDARD RIVISTA+CD €6,90

ioPROGRAMMO

PER ESPERTI E PRINCIPIANTI
Poste Italiane S.p.A Spedizione in A.P. • D.L. 353/2003 (conv.in L.27/02/2004 n.46) art.1 comma 2 DCB ROMA Periodicità mensile •-LUGLIO 2006-• ANNO X, N.7 (104)

Palmari e Smartphone ecco come programmarli!

WINDOWS MOBILE

Progetta subito software per dispositivi che stanno nel palmo di una mano

TEORIA Diversi sistemi operativi per diverse periferiche, quali le differenze?

STRUMENTI Il Compact .NET Framework, installiamolo e usiamolo da Visual Studio

TECNICA Display piccolissimi e dotazioni ridotte. Adattiamo le nostre applicazioni

PRATICA Gestiamo gli ordini verso l'azienda con il Pocket PC

IN PIÙ LA VIDEOGUIDA NEL CD-ROM

msdn WEBCAST
3 NUOVE VIDEOGUIDE PER SVILUPPARE CODICE BLINDATO
SPECIALE SICUREZZA

DATABASE MYSQL ANCORA PIÙ VELOCI

Siti web lenti? ecco le tecniche per capire dove le tue query si "inceppano" ed i consigli per stare sempre al top delle prestazioni

CASI DI STUDIO: JAVA

PAGAMENTI ONLINE

Scegliamo la banca e creiamo un'applicazione con tanto di pos elettronico. Un caso concreto

SQL SERVER 2005

STORED PROCEDURE

Creiamole direttamente in Visual Studio e usiamole come DLL nel database

JAVA

LOGIN SICURO

Sviluppiamo un Web Services per gestire i "Captcha": i codici grafici di confronto per il web

VISUAL BASIC

IL WEB È SERVITO

Impariamo a usare gli Active Document. Form speciali che "vivono" nel browser!

.NET

NON PERDERE MAI IL CODICE!

Utilizza gli "Snippets" e crea piccoli template di procedure riutilizzabili quando servono

CREA LE CLASSI AL VOLO

Impara come funzionano i "Build Compilers" e sviluppa pagine con estensioni mai viste

SOFTWARE DOCUMENTATO

Scrivi applicazioni autoesplicative, per utenti che non devono chiedere mai!

IN PIÙ LA VIDEOGUIDA NEL CD

JAVA

SCRIVILO SUL MURO

Sviluppiamo un GuestBook molto particolare con Spring e il pattern MVC

DA SQL AD OGGETTI

Facciamolo con Cayenne, il tool con interfaccia grafica che rende tutto semplice

FILTRI GRAFICI

Metti un Photoshop nella tua applicazione. Con una libreria semplice e OpenSource

ALGORITMI DI ORDINAMENTO Alberi bilanciati, come inserire, rimuovere e gestire i nodi

Anno X - N.ro 7 (104) - Luglio 2006 - Periodicità Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Fabio Farnesi
Collaboratori: M. Autiero, C. Bellucci, M. Bigatti, R. Brunetti, L. Buono, L. Caputo, D. De Michelis, C. Durante, F. Grimaldi, M. Scala, G. Sudano

Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Art Director: Paolo Cristiano
Responsabile grafico di progetto Salvatore Vuono
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Francesco Cospite

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti
Direttore Generale: Massimo Rizzo

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90 · ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 30/09/06
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Servizio Abbonati:
tel. 02 831212
e-mail: servizioabbonati@edmaster.it

Assistenza tecnica: ioprogrammo@edmaster.it
Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Giugno 2006

Interoperabilità

La parola chiave per il futuro dello sviluppo sembra essere "Interoperabilità". In un mondo in cui convivono architetture, servizi e sistemi differenti appare logico che l'unica strada affinché il mondo "interconnesso" possa continuare a essere utile per i propri utenti è quello di trovare un modo affinché tutti gli attori che lo compongono riescano a comunicare fra loro.
Perciò, Microsoft, Sun, IBM, Oracle e più in basso l'intera community OpenSource spingono per trovare un terreno di intesa comune. Così nasce il consorzio per l'Open Document, così si discute su Soap, così si parla di formati. Ma esattamente qual è la fotografia della situazione? A ben guardare siamo ben lontani da un qualunque accordo. Tutte le varie aziende spingono nella direzione che ritengono più opportuna ed è scontro su tutto. Avete mai provato a sviluppare un Web Services un po' più complesso del solito e magari ad applicarvi delle politiche di security? Se lo avete fatto avrete già scoperto che la tanto mitizzata possibilità di far consumare a un client scritto in un qualunque linguaggio un web services scritto in un altro diventa un'operazione non banale. E vogliamo parlare dei formati? Open Document è attualmente un terreno di scontro durissimo la dove le diverse implementazioni propongono modelli che non possono assumersi in nessun caso l'onere di standard. Cosa possiamo fare noi programmatori? Proprio noi che siamo sempre alla ricerca estrema della tecnica più innovativa, possiamo solo e solamente in questo caso, utilizzare quella meno innovativa ma più consolidata. Perciò se non vogliamo ritrovarci a breve termine con un mondo di software completamente inutilizzabile se non per la piattaforma su cui è stato implementato è il caso di cominciare a fare un passo indietro e per una volta dettare noi al mercato quali sono gli standard che intendiamo adottare. E noi sappiamo che quelli più comodi da adottare sono quelli che facilitano il nostro lavoro. Per una volta dunque, facciamo sentire anche la nostra voce. Suggerimenti, email, considerazioni sono ben accette all'indirizzo *redazione@ioprogrammo.it*, saremo ben lieti di dare visibilità al vostro lavoro.

Fabio Farnesi ffarnesi@edmaster.it

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre \soft\codice\ e \soft\tools\) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

WINDOWS MOBILE

Progetta subito software per dispositivi che stanno nel palmo di una mano

- ✔ **TEORIA: diversi sistemi operativi per diverse periferiche, quali le differenze**
- ✔ **STRUMENTI: Il Compact .net Framework, istalliamolo e usiamolo da Visual Studio**
- ✔ **TECNICA: Display piccolissimi e dotazioni ridotte . Adattiamo le nostre applicazioni**
- ✔ **PRATICA: Gestiamo gli ordini verso l'azienda con il Pocket PC**

pag. 12

DATABASE MYSQL ANCORA PIÙ VELOCI

Siti Web lenti? ecco le tecniche per capire dove le tue query si “inceppano” ed i consigli per stare sempre al top delle prestazioni

pag. 62

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all’articolo e seguire passo passo quanto viene spiegato tenendo d’occhio l’intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell’articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

http://forum.ioprogrammo.it

RIVISTA + CD-ROM in edicola

MySQL SUPER PACKAGE

Gli indispensabili per usare subito il DB più amato dai Web Developer

MYSQL 5.0.21
La nuova versione del Database

MYSQL ADMINISTRATOR 1.1.9
Amministra il server in un click

MYSQL CONNECTOR
Per connettere il Database alle tue applicazioni .NET, Java, J2EE, PHP etc…

MYSQL QUERY BROWSER
Naviga agevolmente fra le tabelle ed esegui query da un'interfaccia grafica comoda ed evoluta

MYSQL MIGRATION TOOLKIT
Per migrare senza problemi da Access, SQL Server, Oracle o altri….

Prodotti del mese

Spring 1.2.8

Il framework che li racchiude tutti

In questo numero lo utilizziamo per costruire un guestbook piuttosto particolare. Nell'articolo del bravo Ivan Venuti si mostra come implementare il pattern MVC con Spring. Si tratta di un framework che sta ottenendo un grande successo, per due motivi fondamentali. Prima di tutto raccoglie sotto un unico cappello una serie di tecnologie già esistenti, donandogli però una certa interoperabilità e una forma omogenea. Secondo, implementa perfettamente il pattern Inversion Of Control che consente un alto grado di disaccoppiamento tra le classi di modo che la modifica di una non cambi il comportamento di molte altre. Si tratta di un pattern molto interessante, per quanto sia investito di una certa complessità. Tuttavia non mancheremo di supportarvi nello sviluppo.

[pag.106]

Ruby 1.8.2

Il nuovo che avanza

E' così quest'anno il premio come miglior prodotto dell'anno va a RubyOnRail un framework basato su Ruby. Quando qualche anno fa questo linguaggio fece la sua comparsa sul mercato, nessuno gli diede molto credito. Ad oggi si presenta come un linguaggio di scripting eccezionale che condensa in un'unica soluzione un modello ad oggetti completo, una curva di apprendimento che rasenta quasi lo zero assoluto, un alta integrazione con gli oggetti com di windows, e altre elementi ancora... per creare un web services completo sono sufficienti due righe. In questo numero mostriamo come creare grafici animati in excel utilizzando appunto Ruby

[pag.106]

Ruby on Rails

Il framework vincitore del premio "best tool" del 2006

Chi lo avrebbe mai detto che un framework basato su Ruby avrebbe vinto un premio così prestigioso? Ed invece ad anni di distanza dalla presentazione di Ruby, ecco arrivare un premio che lo consacra fra i linguaggi più interessanti della rete. Ruby On Rail è un framework per lo sviluppo di applicazioni Web, solide e affidabili. Soprattutto la bassa curva di apprendimento di Ruby e l'ottima progettazione del framework ne fanno un'accoppiata molto interessante per lo sviluppo di applicazioni Web. Proprio per questi motivi Ruby On Rail si presenta come uno strumento da apprendere. Certamente lo sforzo iniziale non saranno elevati, al contrario la velocità di sviluppo ne trarrà enormi benefici

[pag.107]

YetAnotherForum 1.0.1

Ancora un altro forum!

Il gioco di parole si presta bene per marcare la vocazione OpenSource di questo progetto. Di fatto, tipicamente questo genere di gioco di parole viene applicato proprio a software OpenSource. YetAnotherForum è un forum scritto interamente in .NET. Il progetto è piuttosto ambizioso e si pone sulla scia di progetti ben più noti quali phpBB. Tuttavia è importante notare oltre alle qualità intrinseche di questo forum che sono veramente eccellenti, quanto anche in ambienti .NET si stia cominciando a diffondere il concetto di OpenSource. Siamo convinti vedremo yetanotherforum in versioni riviste e migliorate proprio grazie all'apporto della community.

[pag.107]

I contenuti del libro

Imparare C#

Oggetti, classi, polimorfismo. Si tratta dei concetti base che rendono la programmazione moderna affascinante e al contempo commisurata alle esigenze di una società che necessità sempre più di "rapidità". Ma quali sono i linguaggi che più di altri interpretano in modo ottimale questi concetti? C# sicuramente è uno di questi. Nato in stretta congiunzione alla tecnologia .NET non solo ne eredita tutti i vantaggi ma vi accoppia una sintassi e una semantica che sono propri di un linguaggio evoluto e perfettamente aderente alle logiche richieste ad un moderno linguaggio ad oggetti. Michele Locuratolo con un linguaggio semplice e un'informazione sempre precisa ci conduce dalle basi fino alla trattazione di elementi complessi che possono essere utili anche ai programmatori più esperti

IL LINGUAGGIO PER I PROGRAMMATORI PROFESSIONISTI E PER CHI ASPIRA A DIVENTARLO

- Elementi del linguaggio
- Classi ed oggetti
- Array indici e collections
- Novità di C# 2.0

GLI ALLEGATI DI IOPROGRAMMO

Applicazioni sicure

L'informatica moderna ha portato alla ribalta alcuni temi che solo fino a qualche tempo fa sembravano esistere solo in uno scenario da fantascienza. Oggi un programmatore deve tenere in conto la necessità di sviluppare applicazioni sicure.
In queste aree Microsoft agisce con un triplice impegno. Dal punto di vista del produttore di applicazioni, dal punto di vista del produttore del sistema operativo, dal punto di vista di chi mette a disposizione framework evoluti per la scrittura di applicazioni. In questa serie di tre Webcast MSDN vengono illustrate alcune delle principali tecniche di sicurezza. Si parte da una descrizione dei meccanismi di protezione offerti dal kernel, per poi passare ad approfondire come il sistema operativo gestisce la sicurezza per i servizi e per gli utenti che li eseguono, terminando con la presentazione delle politiche da adottare per la riduzione dei privilegi assegnati agli utenti. Nonostante la sicurezza del sistema, non dimentichiamoci, però, che buona parte dei bug di sicurezza dipendono da come noi scriviamo le applicazioni.

I VIDEOCORSI PER PROGRAMMARE BENE

3 WEBCAST UFFICIALI MICROSOFT

IN ESCLUSIVA GRATIS NEL CD "I CORSI DI FORMAZIONE"
DA SEGUIRE COMODAMENTE SUL PC

msdn WEBCAST

- **Windows: la sicurezza inizia dal kernel**
- **Il processo di logon e la sicurezza per l'utente interattivo e per i servizi**
- **La riduzione dei privilegi**

INFORMAZIONI SU MSDN WEBCAST
http://www.microsoft.it/msdn/webcast_msdn
http://forum.ioprogrammo.it

FAQ

Cosa sono i Webcast MSDN?

MSDN propone agli sviluppatori una serie di eventi gratuiti online e interattivi, che approfondiscono le principali tematiche relative allo sviluppo di applicazioni su tecnologia Microsoft. Questa serie di "corsi" sono noti con il nome di Webcast MSDN.

Come è composto tipicamente un Webcast?

Normalmente viene presentata una serie di slide commentate da un esperto di tecnologie Microsoft. A supporto di queste presentazioni spesso vengono realizzate delle demo in presa diretta che mostrano dal vivo come usare le tecnologie oggetto del Webcast

Come mai nei webcast allegati alla rivista si parla di chat e di altri strumenti interattivi?

La natura dei Webcast è quella di essere seguiti online e in tempo reale. Durante queste presentazioni in diretta vengono utilizzati strumenti molto simili a quelli della formazione a distanza. E' possibile porre domande in presa diretta al relatore oppure partecipare a sondaggi che vengono proposti. In questo modo gli sviluppatori possono aggiornarsi e approfondire i temi di loro interesse con maggiore efficacia. I Webcast riprodotti nel CD di ioProgrammo, pur non perdendo nessun contenuto informativo, per la natura asincrona del supporto non possono godere dell'interazione diretta con il relatore.

Come mai trovo i WebCast su ioProgrammo

Come sempre ioProgrammo cerca di fornire un servizio ai programmatori italiani. Abbiamo pensato che poter usufruire dei Webcast MSDN direttamente da un CD rappresentasse un ottimo modo di formarsi comodamente a casa e nei tempi desiderati. Lo scopo tanto di ioProgrammo, quanto di Microsoft è infatti quello di supportare la comunità dei programmatori italiani con la più ampia gamma di strumenti di formazione e aggiornamento.

Su ioProgrammo troverò tutti Webcast MSDN?

Ne troverai sicuramente una buona parte. Direttamente sul sito MSDN potrai consultare il calendario dei prossimi webcast a cui iscriverti per seguirli in diretta oppure consultare l'archivio delle registrazioni di quelli già realizzati. L'indirizzo per saperne di più è http://www.microsoft.it/msdn/webcast_msdn, segnalo nel tuo bookmark, non può mancare!

L'iniziativa sarà ripetuta sui prossimi numeri?

Sicuramente sì, alla prossima.

News

JAVA FA GIRARE VISUAL BASIC

Non si tratta di un paradosso, ma del progetto Visual Basic for the Java Platform , presentato anche esso durante la scorsa JavaOne. Lo scopo è semplice: consentire ad applicazioni scritte in VB di girare tramite la piattaforma Java. Per certi versi Java ripercorre la strada effettuata da .NET, ovvero metterà a disposizione un compilatore che consentirà di ottenere un Byte Code Java a partire da sorgenti Visual Basic. Ora, se si pensa che Java già molto prima di .NET aveva nelle corde questo tipo di possibilità, ci si accorge di quanto tempo abbiano perso i progettisti di Sun o forse di quanto lungimiranti siano stati quelli di Microsoft. Resta il fatto che .NET proprio grazie alla possibilità di far girare tutta una serie di linguaggi ricompilando il sorgente in una sorte di Byte Code sta ottenendo in tempi brevi un successo enorme. D'altra parte Java che è nato più di 10 anni fa non ha colto immediatamente questa opportunità. Vedremo se nel tempo i progettisti di Sun sapranno recuperare il terreno perduto.

ORACLE RILASCIA UNA NUOVA VERSIONE DI BERKELEY DB

Sono passati pochi mesi da quando Oracle ha acquistato Berkeley DB ed ecco arrivare immediatamente una nuova versione. Le innovazioni riguarderebbero una serie di interfacce API messe a disposizione dei programmatori Java e un nuovo motore di persistenza. Se si considera che Berkeley DB è uno dei primi software di database ad essersi affacciato sul mercato e se si considera la sua natura opensource, appare chiara la volontà di Oracle di affacciarsi su un mercato, ovvero quello delle piccole e medie aziende, che fino ad ora gli era sfuggito totalmente. Sia il rilascio di una versione Express di Oracle 10g sia il rilascio di questa nuova versione di Berkeley DB lasciano intuire quanto sia appetibile il mercato delle piccole e medie imprese, soprattutto per i produttori di software database oriented

ARRIVA IL TECHNET SECUR

Sesta edizione per il Microsoft Technet Security Workshop. Si è tenuto a Milano il 13 e il 15 Giugno il consueto appuntamento con i professionisti dell'Information Technology sui temi della sicurezza. Quest'anno l'intero workshop ha focalizzato la propria attenzione sulle tecnologie e I protocolli su cui si basano I certificati di autenticazione e sulle modalità con cui possono essere utilizzati. "Ci rivolgiamo ai professionisti IT per affrontare uno dei temi che da anni ci é più a cuore: la sicurezza delle infrastrutture e dei sistemi. Così è nato sei anni fa il TechNet Security Workshop", aveva dichiarato Raffaella Verticchio, TechNet Program Manager di Microsoft Italia. "L'obiettivo è fornire a tutti coloro che si occupano della gestione dei sistemi IT nelle aziende italiane momenti di aggiornamento e confronto sulle nuove tecnologie e sui nuovi prodotti in grado di incrementare il livello di sicurezza in azienda. In particolare, quest'anno parleremo di certificati digitali, l'elemento chiave delle moderne tecnologie informatiche in tale ambito". Particolare attenzione dunque ai certificati digitali e alle tecnologie ad essi correlati, segno che il traffico di rete sta diventando sempre più la catena debole su cui concentrare l'attenzione se si vogliono evitare problemi seri di sicurezza. In una società in cui la nostra intera identità può sempre in una qualche misura risiedere in rete è importante capire come si possa concentrare la propria attenzione sulle parti che compongono una transazione, consumer e server. In tutto questo non

JPEG AVRÀ UN EREDE

Si chiama WMPhoto, ovvero Windows Media PHOTO ed ha fatto capolino sul mercato durante la recente Windows Hardware Engineering Conference. A presentarlo è stata Microsoft che ha annunciato di voler includere il nuovo formato nel nascituro Windows Vista. Secondo la casa di Redmond, il nuovo formato ha tutte le carte in regola per soppiantare l'onnipresente JPEG. L'algoritmo di compressione al suo interno è tale da garantire, sempre secondo Microsoft, dimensioni dimezzate rispetto alla normale compressione JPEG. Il Windows Media Photo si pone nell'agguerrito gruppo dei formati di compressione elaborato da Microsoft ovvero Windows Media Video e Windows Media Audio. Se le aspettative fossero rispettate, il nuovo formato diverrebbe in breve tempo uno standard per quanto riguarda i dispositivi mobili. Certamente WMPhoto potrebbe risultare utile anche sui dispositivi standalone, ma data la crescente disponibilità di periferiche di storage capaci di contenere enormi moli di dati a prezzi contenuti, probabilmente non ci sarebbe in questo senso una grande rivoluzione. Allo stesso modo l'aumento della larghezza di banda fa si che la dimensione delle foto non sia più un problema anche sul Web. Viceversa nei dispositivi mobili questo tipo di formato troverebbe una collocazione ideale. E se si pensa che il nuovo Windows Mobile 5.0 sta ormai scalando molte posizioni nella classifica dei sistemi operativi per sistemi mobili, c'è da scommettere che WMPhoto otterrà sicuramente un buon successo.

ITY WORKSHOP

poteva non mancare uno sguardo attento alle tecnologie WiFi e alle problematiche di sicurezza che queste ultime hanno messo in luce. In una società in continua evoluzione, in cui palmari, smartphone e altri dispositivi rendono mobile persino il concetto di rete aziendale, diventa importante proteggere le proprie credenziali di autenticazione e i propri dati. Moltissimi sforzi devono essere fatti per "educare" utenti e sistemisti ad una nuova logica di gestione che sta irrompendo velocemente all'interno di abitudini consolidate, non lasciando il tempo ai gestori di adeguarsi, ma molto deve essere fatto anche in termini di progettazione di applicazioni sicure che tengono conto delle sempre possibili incertezze dell'utente inesperto e della contrapposta furbizia di chi invece studia la rete in tutti i suoi meccanismi più per ottenere un vantaggio da eventuali buchi lasciati scoperti

TRE BETA PER MICROSOFT

Windows Vista, Office 2005 e Longhorn Server. Sono queste le tre nuove versioni Beta annunciate a "reti unificate" da Microsoft durante la recente Windows Hardware Engineering Conference. Il rilascio di queste tre nuove versioni in modo pressoché contemporaneo è sinonimo secondo Microsoft di sinergia e collaborazione. Di fatto questi tre software costituiscono una piattaforma operativa all'interno della quale si muovono in perfetta sincronia tutti gli attori di un'ottima infrastruttura informatica. I tre pilastri di questa architettura sono costruiti per interagire in modo stretto fra di loro favorendo una maggiore usabilità da parte degli utenti. Gli obiettivi sui quali Microsoft si sarebbe concentrata sono quattro: garantire una piattaforma di collaborazione semplice fra le persone, migliorare la protezione dei contenuti, offrire degli strumenti di ricerca evoluti, migliorare la sicurezza generale. C'è soddisfazione in casa Microsoft per questo annuncio, ma si è anche immediatamente aperto il fronte delle polemiche. Longhorn integrerebbe infatti una piattaforma di virtualizzazione piuttosto evoluta, e nelle parole di J. Woolsey , lead program manager della divisione Windows di Microsoft , migliore di qualunque altro concorrente. Non si è fatta attendere la risposta di VMware che per bocca di uno dei suo vicepresidenti: Raghu Raghuram Raghuram, ha dichiarato VMWare sarà subito in grado di sfruttare tutte le migliorie offerte da Longhorn

PROGRAMMA E VINCI UNA XBOX 360

E' il premio messo in palio da Microsoft alla prima shared source programming contest. I partecipanti possono scaricare una trial version di Windows CE che rimarrà valida per 120 giorni e tutta una serie di strumenti di sviluppo ad essa correlati. In cambio dovranno produrre software di una qualche utilità nel mondo reale. I giudici premieranno l'originalità, la documentazione, la reale utilità, e porranno una particolare attenzione ad un piccolo video dimostrativo che dovrà essere realizzato dai partecipanti. In palio c'è un pacchetto contenente una XBOX 360, un televisore da 34 pollici e una marea di software per un totale di circa 2500$, sono ovviamente previsti anche premi minori. Le iscrizioni scadranno il 28 luglio e il vincitore sarà annunciato intorno alla metà di Agosto. L'entusiasm è già alto e le iscrizioni procedono a ritmo incessante.

Chi volesse partecipare può trovare tutte le informazioni all'indirizzo http://www.windowsfordevices.com/articles/AT5277795134.html

IN ARRIVO SOA 2.0

Applicazioni basate su servizi e che comunicano fra loro tramite messaggi, in una parola Service Oriented Application, ovvero SOA. E' questa la parola chiave del momento, la meta verso cui tutte le aziende produttrici di software si dirigono. Così Oracle ha deciso di bruciare i tempi e presentare durante la scorsa JavaOne la sua nuova versione di SOA. La nuova architettura metterebbe in relazione la programmazione basata su servizi e quella basata su eventi.

"SOA 2.0 è il temine che usiamo per parlare della combinazione della combinazione di architetture service-oriented e architetture eventdirven" questo è quanto ha dichiarato "Steve Harris" vice presidente di Oracle Fusion Middleware. Allo stesso modo Yefim Natis ha specificato: "SOA così come lo conosciamo oggi propone un'architettura client server fra moduli software con i servizi che diventano una sorta di subroutine dei client", l'architettura proposta da SOA 2.0 invece prevede che i servizi possano lanciare alert o eventi di notifica verso il client, diventano in questo modo elementi attivi dell'architettura.

Come partner per lo sviluppo di SOA 2.0, Oracle ha individuato la piattaforma Java Enterprise 5, inoltre è molto probabile che in tutta l'architettura una parte dominante l'avrà Ajax ovvero la tecnologia di tunneling http su cui si sta basando buona parte del WEB moderno. Il tool di sviluppo principe su cui sarà incentrata l'intera piattaforma sarà JDeveloper,

SVILUPPO PER DISPOSITIVI MOBILI

PROGETTIAMO UNA COMPLETA APPLICAZIONE ED ENTRIAMO NELLA LOGICA DI FUNZIONAMENTO DELLA NUOVA VERSIONE DEL COMPACT FRAMEWORK E DI WINDOWS MOBILE 5.0. USIAMO VISUAL STUDIO 2005 PER RENDERE TUTTO PIÙ SEMPLICE

L'informatica ha letteralmente cambiato il nostro modo di lavorare. Le comunicazioni sono più rapide, le distanze si sono ridotte, il lavoro ripetitivo è stato pressoché eliminato, la produttività è aumentata. Molte sono le tecnologie che, insieme, hanno contribuito a questo radicale cambiamento del nostro modo di lavorare. Si va dalla potenza di calcolo dei computer notevolmente aumentata negli ultimi anni, alla connettività sempre più presente ed economica.
Uno degli elementi chiave che però ha contribuito maggiormente a modificare il nostro modo di lavorare è il concetto di Mobility. "Business Everyware" è uno dei tanti slogan che si legge durante la presentazione di tutto ciò rientra nel concetto di mobility. Slogan che, a mio parere, rende perfettamente l'idea di quello che stà accadendo.
Se di recente avete preso un aereo, sapete a cosa mi riferisco.
Le sale d'attesa sono cambiate e, le persone sedute a leggere il giornale sono state pian piano sostituite da persone con un PC portatile appoggiato sulle ginocchia che magari stanno rispondendo ad una mail o stanno inviando un ordine in azienda.
Personalmente, grazie ad uno smartphone ed una connessione GPRS, non raramente mi capita di rispondere a qualche mail mentre sono bloccato nel traffico.
Il nostro modo di lavorare stà cambiando, che ci piaccia o no. Quello che possiamo fare, come sviluppatori, è cogliere le opportunità che questi cambiamenti ci riservano, producendo software adeguati. In questo articolo, dopo una panoramica sui dispositivi mobili in generale, passeremo alle novità introdotte dal nuovo sistema operativo Microsoft denominato Windows Mobile 5.0 e la nuova versione del Compact Framework (2.0). Grazie ad essi, realizzeremo poi una applicazione per la gestione delle spedizioni.
Mettiamoci subito all'opera.

I DISPOSITIVI MOBILI

La scelta di dispositivi mobili oggi è, fortunatamente, abbastanza vasta. Per ogni particolare esigenza, ci sono a disposizione numerosi dispositivi, diversi per caratteristiche hardware, funzionalità e, cosa non meno importante, il costo.
Principalmente, le tipologie di dispostivi mobili, sono 4:

- Dispositivi industriali.
- Pocket PC.
- Pocket PC con funzionalità telefoniche.
- Smartphone.

Nella prima categoria rientrano tutti i dispositivi caratterizzati principalmente da una elevata robustezza. Il loro ambito di utilizzo è generalmente quello industriale, su cantieri o magazzini e, di conseguenza, devono essere resistenti agli urti, alla polvere ed all'umidità.
Generalmente questo tipo di dispositivi è abbastanza costoso, ma sono indispensabili nei contesti sopra elencati. Il secondo tipo di dispositivi invece è caratterizzato da un amplio display, dal supporto all'input con la penna, dal display touch screen e, sempre più spesso, da una piccola tastiera querty integrata.La terza categoria è identica alla precedente a cui però si aggiungono le funzionalità telefoniche. Gli Smartphone invece, sono molto simili a dei normali telefoni cellulari. Caratterizzati da un display più piccolo dei Pocket PC, non sono dotati ne di touch screen ne di tastiera, se non quella numerica classica. Sono molto comodi per controllare la posta, rispondere a brevi messaggi, avere con se una rubrica ed una agenda sincronizzabili con Microsoft Outlook ed, ovviamente, per telefonare. La loro natura però li rende decisamente scomodi per navigare o qualora si abbia la necessità di scrivere parecchio.
Quando si sceglie un dispositivo, bisogna considerare innanzitutto l'uso che se ne deve

fare. Se il nostro obiettivo principale è quello di poter sempre ricevere la posta elettronica e gestire i nostri appuntamenti, la scelta più opportuna sarò uno Smartphone (piccolo e comodo da trasportare). Se al contrario siamo dei rappresentanti, un Pocket PC (magari Phone Edition) sarebbe la scelta più sensata.

Windows Mobile 5.0
Windows Mobile 5.0 è l'ultimo sistema operativo Microsoft per dispositivi mobili. Ne esitono 3 versioni principali, ognuna adatta ad un particolare dispositivo mobile:

- **Pocket PC Edition:** è la versione per dispositivi Pocket PC privi di funzionalità telefoniche.
- **Pocket PC Phone Edition:** è la versione per dispositivi dotati di funzionalità telefoniche
- **Smartphone Edition:** è la versione per dispositivi Smartphone

Ognuna delle versioni di Windows Mobile 5.0 elencate, ha funzionalità specifiche per il tipo di device per cui è stata progettata. Nella versione Smartphone ad esempio, mancheranno il supporto per la tastiera e per l'input tramite penna. Quando dobbiamo sviluppare una applicazione per dispositivi mobile, dobbiamo tenere conto del device in su cui essa dovrà essere istallata. Questo per non incorrere in problemi dovuti all'assenza di supporto per determinate funzionalità.
Se ad esempio dobbiamo sviluppare un software per la gestione delle spedizioni (come quello presentato in queste pagine), dobbiamo considerare il fatto che avremo la necessità di inserire e visualizzare correttamente dei dati. La scelta di uno smartphone come target, potrebbe non essere la più sensata. Windows Mobile 5.0 introduce numerose novità, non tutte elencabili in queste pagine. Ci concentreremo quindi principalmente su quelle che riguardano lo sviluppo di applicazioni.Il punto centrale del rinnovamento di Windows Mobile 5.0 per gli sviluppatori è l'aumento della produttività. Innanzitutto è stato aumentata la compatibilità tra le applicazioni Pocket PC e Smartphone rendendo di fatto meno dolorosa la loro migrazione (resta inteso che sarebbe comunque opportuno scegliere adeguatamente la piattaforma).
L'aumento della produttività è anche dovuto ad una serie di nuove API che espongono al mondo managed, molte delle funzionalità del sistema operativo. Tra le più significative ci sono indubbiamente:

- **Funzionalità di telefonia:** che permettono di interagire con le funzionalità telefoniche dei sistemi Phone Edition e Smartphone
- **Outlook Mobile:** che semplificano notevolmente l'uso delle funzionalità di Outlook Mobile attraverso applicazioni managed
- **Configuration Manager:** fornisce la possibilità di impostare numerosi parametri di configurazione del device attraverso un file xml.
- **Uso della fotocamera:** le funzionalità di cattura immagini e video della fotocamera sono ora esposte. E' quindi possibile, con estrema semplicità, dotare le nostre applicazioni di funzionalità legate all'uso di questo dispositivo.
- **GPS:** rappresentano un set di funzionalità intermedio che permettono di accedere con semplicità ai dati GPS. Particolarmente utili quando si ha la necessità di applicazioni che usano ad esempio MapPoint.
- **Integrazione di active Sync:** è ora possibile avviare o sospendere via codice la sincronizzazione dei dati via Active Sync

L'insieme di queste funzionalità native e del nuovo Microsoft .NET Compact Framework, rendono possibile lo sviluppo di applicazioni complesse e ricche di funzionalità molto più semplicemente che in passato.

I TUOI APPUNTI

LE NOVITÀ DEL COMPACT FRAMEWORK 2.0

Recentemente, Microsoft ha rilasciato la nuova versione del .NET Framework (la 2.0) che include anche la versione per dispositivi mobili denominata Compact Framework.
Questa versione, dato il tipo di dispositivi su cui deve essere installata ed eseguita, è chiaramente una versione ridotta del framework installabile sui tradizionali PC. Non includa tutti i namespace e tutte le funzionalità del fratello maggiore, il Compact Framework permette di sviluppare applicazioni anche decisamente complesse per Pocket PC e Smartphone. In particolare, questa nuova versione, introduce numerose funzionalità sia per quanto riguarda la migliorata compatibilità con il tradizionale framework, sia per novità nello sviluppo di interfacce utente che di integrazione con il sistema operativo.
In questo paragrafo, analizzeremo alcune tra le novità introdotte dalla versione 2.0. Per ovvie esigenze di spazio, verranno tralasciate quelle meno importanti, comunque consulta-

bili on line all'indirizzo riportato nel box laterale.
Iniziamo dall'interfaccia grafica:

- **Notification:** sfrutta il sistema di notifiche di Windows Mobile per segnalare eventi con dei baloon (stile Windows XP). Oltre alla semplice visualizzazione, questo nuovo controllo permette di gestire un eventuale input di un utente.

- **DocumentList:** permette di navigare nel file system del dispositivo per scegliere, copiare, eliminare dei files.

- **DataGrid per Smartphone:** nella precedente versione del .NET Compact framework, questo controllo era disponibile solo per Pocket PC. Da oggi è disponibile anche per SmartPhone.

- **Splitter e Toolbar:** sono state aggiunte anche sul Compact Framework.

- **User Control:** come per il .NET Framework, ora sarà possibile sviluppare user control anche per dispositivi mobili. Tali user control potranno essere inseriti nella toolbox ed usati in altri progetti.

- **Display and Layout:** in questa categoria rientrano una serie di nuove caratteristiche utili a migliorare la gestione dell'interfaccia come control docking, control anchoring, automatic scrolling etc.

Fig. 1: ***La comparazione tra le toolbar delle due versioni di Visual Studio***

Per rendersi immediatamente conto della quantità di controlli aggiunti in questa nuova versione, possiamo paragonare le toolbox di Microsoft Visual Studio .NET 2003 e Microsoft Visual Studio .NET 2005. In Figura 1, sono riportate entrambe affiancate.
La maggior parte delle applicazioni oggi, fa uso di un Data Base in cui archiviare i dati. Le applicazioni per dispositivi mobili non fanno eccezione. Ecco quindi alcune novità introdotte dal nuovo framework a tal proposito:

- **SqlCeResultSet:** fornisce un accesso al Data base Sql Server 2005 Mobile Edition molto più veloce e leggero di un Data Set. SqlCeResultSet può essere usato come Binding Source per i nostri controlli.

- **DataSet:** è stato introdotto anche nel Compact Framework questo comodo componente per la gestione disconnessa dei dati.

Tra le altre funzionalità aggiunte, è importante sottolineare le seguenti:

- Supporto migliorato per XML

- Web Service più performati grazie alle migliorie fatte all'XMLSerializer

- **Serial Port:** introdotto nativamente in questa versione del Compact Framework, semplifica notevolmente lo sviluppo di applicazioni che hanno necessità di interfacciarsi con dispositivi seriali come lettori di codici a barre o RFID.

- Supporto migliorato per la crittografia

SUL WEB

Tutte le novità relative al Microsoft .NET Compact Framework 2.0, sono consultabili al seguente indirizzo web:
http://msdn.microsoft.com/netframework/programming/netcf/default.aspx?pull=/library/en-us/dnnetcomp/html/whats_new_netcf2.asp .

- **Threading:** introduce la possibilità di sviluppare applicazioni multithreading anche su dispositivi mobili. Tale funzionalità è decisamente importante al fine di sviluppare applicazioni che non blocchino l'interfaccia utente in attesa del completamento delle operazioni.

- **Performance:** in aggiunta a quanto fin ora commentato, nel .NET Compact Framework sono stati introdotti significativi incrementi di performance che abbracciano sostanzialmente tutti i campi di applicazione.

Dopo aver visto le novità di Windows Mobile 5.0 e del .NET Compact framework 2.0, iniziamo con lo sviluppo della nostra prima applicazione, ma non senza prima aver fatto qualche doverosa considerazione sullo sviluppo su dispositivi mobili.

CONSIDERAZIONI SULLO SVILUPPO

Prima di iniziare con lo sviluppo di applicazioni per questo tipo di dispositivi, è bene fare alcune considerazioni importanti per evitare errori di progettazione che potrebbero portarci, un domani, a dover rivedere l'intera applicazione.
La prima cosa da considerare quando ci si approccia allo sviluppo di una applicazione per dispositivi mobili è la dimensione dei display. Mediamente, lo spazio a diposizione è di 250x250 pixel (dall'area totale a disposizione devono essere esclusi lo spazio per la barra superiore e quello dei menù in basso.
In questo spazio decisamente limitato, dovranno trovarsi tutte le informazioni più utili per svolgere il task specifico per cui quel form è stato realizzato.
Altra cosa da tenere sempre a mente è che, chi utilizzerà la nostra applicazione, non siederà comodamente di fronte ad un PC con ampio display, tastiera e mouse ma, nella maggior parte dei casi avrà il device in una mano e dovrà interagire con la nostra applicazione con il pennino. E' quindi sconsigliabile avvicinare troppo tra loro i controlli che necessitano di una interazione con l'utente, come bottoni, caselle di input etc. Molto importante risulta il posizionamento dei controlli, specie per quelli che richiedono una compilazione attraverso il pennino o la tastiera su schermo.
Il panel di input infatti, normalmente nascosto, potrebbe coprire i controlli che stiamo compilando, rendendo molto scomoda la compilazione. Vediamo subito un esempio di quello che NON si dovrebbe fare, in figura 2. Le scelte di design in questo tipo di progetto sono fondamentali.
Tra le altre cose da considerare, non è da sottovalutare la potenza di calcolo ridotta di questi dispositivi. Oltre alle ottimizzazioni necessarie da fare sul codice, va mostrato all'utente che l'applicazione non è bloccata, ma stà lavorando al completamento di un determinato task.
La cosa è fattibile usando controlli come la progress bar o modificando il cursore impostandolo a WaitCursor.

Fig. 2: ***Il posizionamento dei controlli su applicazioni per PPC.***

SUL WEB

Il slto web Microsoft per sviluppatori, dedicato al mondo dei dispositivi mobili, è consultabile al seguente indirizzo: http://msdn.microsoft.com/mobility/ Partendo dalla pagina iniziale, si po' avere accesso a tutte le informazioni utili allo sviluppo per dispositivi mobili.

SINCRONIZZAZIONE DEI DATI

la sincronizzazione dei dati tra un dispositivo mobile ed un server centrale può avvenire in 3 modi distinti: usando le repliche di SQL Server, usando RDA (Remote Data Access) ed usando un servizio web. Le repliche sono molto potenti e consentono una dettagliata gestione dei conflitti, cosa frequente quando si lavora con utenti disconnessi. Hanno però lo svantaggio di essere complesse da configurare e, in alcuni casi, da gestire. Remote Data Access è più semplice da configurare ma ha un sistema poco evoluto per la gestione dei conflitti. L'ultimo, il Web Service, lascia agli sviluppatori il compito di gestire l'intero processo di sincronizzazione nonché di gestione dei conflitti, ma ha il vantaggio di non dipendere dal Data Base. Al seguente indirizzo web http://msdn2.microsoft.com/en-us/library/ms172916.aspx, è disponibile una comparazione tra RDA e Repliche.

L'APPLICAZIONE

Dopo aver analizzato le varie novità introdotte da Windows Mobile 5.0 e dal Micorsoft .NET Compact Framework 2.0, realizziamo una applicazione d'esempio che metta insieme un po' delle novità appena viste.
Come tipo di applicazione si è scelto di realizzare un classico programma per la creazione di ordini, usato generalmente dai rappresentanti. Il programma fa uso di un Data Base SQL Server 2005 Mobile Edition locale per la memorizzazione dei dati che poi potrà essere eventualmente sincronizzato con un Data Base centrale.
L'argomento della sincronizzazione, data la sua vastità e complessità (specie in merito alla gestione dei conflitti), non verrà trattato in questo articolo. Verranno comunque forniti dei riferimenti alla documentazione on line. Per lo sviluppo di questa applicazione di esempio, sono stati usate alcune delle novità di maggior rilievo introdotte dal Compact Framework e da Windows Mobile 5.0. Useremo i nuovi BindingSource, SQL Server 2005 Mobile edition, Notifications e due funzionalità per comunicare con il sistema operativo: inserire un appuntamento in Pocket Outlook e chiamare telefonicamente il cliente recuperando il numero di telefono dal Data Base. Mettiamoci all'opera.

SUL WEB

Il Microsoft Windows Mobile 5.0 SDK è scaricabile al seguente indirizzo web: http://www.microsoft.com/downloads/details.aspx?familyid=83A52AF2-F524-4EC5-9155-717CBE5D25ED&displaylang=en **ed include gli emulatori ed i template di Microsoft Visual Studio .NET 2005.**

IMPOSTIAMO IL PROGETTO

Dovendo sviluppare un sofware per Windows Mobile 5.0, è necessario scaricare il relativo Software Development Kit dal sito web di Microsoft (vedi box laterale). Una volta installato, sarà sufficiente creare una nuova applicazione per Smart Device in Microsoft Visual Studio .NET 2005 scegliendo come target naturalmente Windows Mobile 5.0

Fig. 3: ***Creazione del progetto in Microsoft Visual Studio .NET 2005***

Una volta creato il progetto, dobbiamo scegliere su quale tipo di device deve essere testato. Selezioniamo dall'elenco Windows Mobile 5.0 Pocket

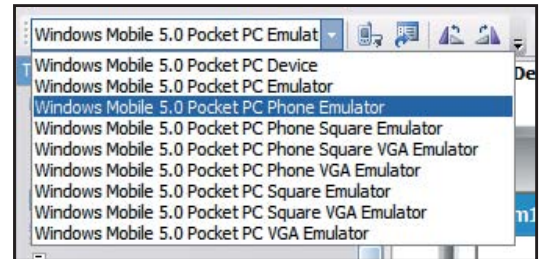



Fig. 4: ***Scelta del Device***

Infine aggiungiamo i riferimenti alle API di Windows Mobile 5.0, in particolare a PocketOutlook e Telephony

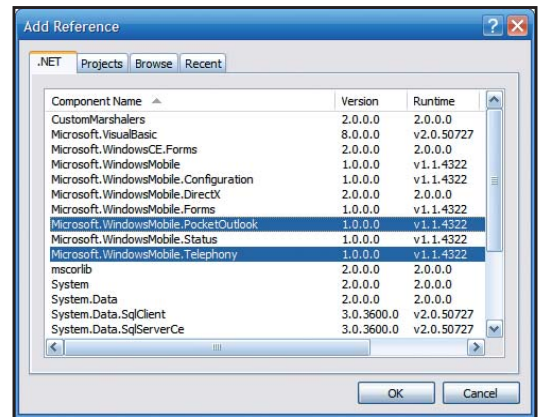



Fig. 5: ***Aggiunta dei riferimenti***

Una volta predisposto il progetto, dobbiamo aggiungere il Data Base che sarà utilizzato per la memorizzazione dei dati. Chi ha lavorato con Microsoft Visual Studio .NET 2003 su applicazioni per dispositivi mobili, ricorderà quanto questa operazione fosse poco pratica. Il Data Base andava creato direttamente sul dispositivo o usando lo scomodo Query Analyzer installato, o creandolo via codice. In ogni caso, dall'interno dell'ide, non era possi-

Fig. 6: ***Aggiunta del Data Base al progetto.***

bile lavorare sul Data Base.
Questa limitazione è finalmente stata eliminata con questa versione. Sarà quindi sufficiente aggiungere al progetto un nuovo elemento e selezionare la voce SQL Mobile DataBase

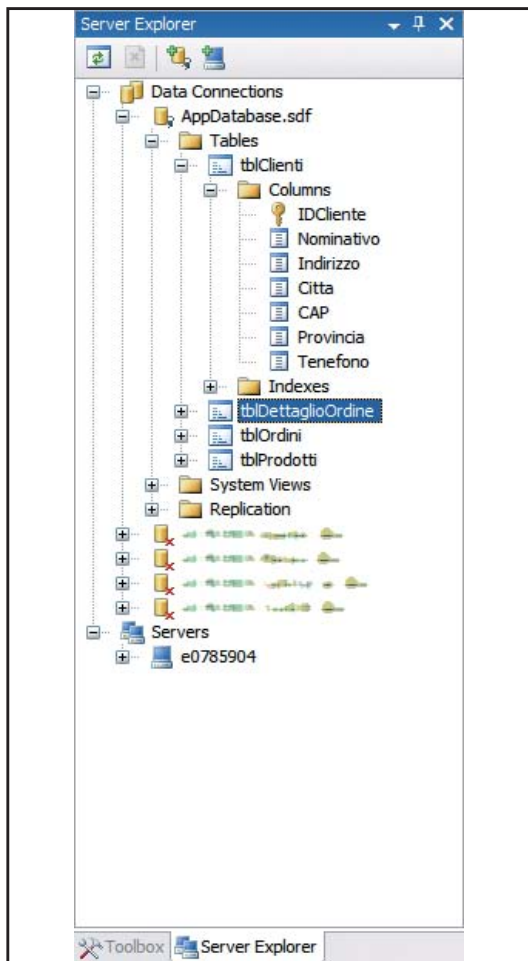



Fig. 7: ***Gestione del Data Base dal panel Server Explorer.***

Il Data Base sarà poi facilmente gestibile attraverso il panel Server Explorer di Microsoft Visual Studio .NET 2005.
L'aggiunta di un Data Base, comporterà la creazione automatica di un Data Set che sarà la nostra sorgente di dati in tutta l'applicazione. Sebbene avessimo potuto interagire direttamente con il Data Base, l'utilizzo del Data Set rende estremamente rapido l'associazione dei dati ai controlli delle nostre interfacce.

BINDINGSOURCE

Il componente BindingSource rientra nelle novità del Compact Framework 2.0 e più in generale, del Microsoft .NET Framework stesso. Lo scopo di questo componente è quello di semplificare l'associazione dei controlli alla base dati. E' possibile usare questo componente per creare associazioni semplici come ad esempio una textBox ad un campo specifico del Data Base o associazioni più complesse come ad esempio una tabella di un Data Base ad una Data Grid.
Nel nostro caso specifico, l'oggetto Binding Source si rivelerà molto comodo per entrambe le esigenze, soprattutto per creare form in cui i controlli si aggiornano sulla base di un record selezionato nella Data Grid.
Aggiungere un BindingSource al form è una operazione estremamente rapida. Ci basterà trascinarlo sulla form e configurarne l'associazione ai dati.
Vediamone un esempio pratico nella foto in basso.

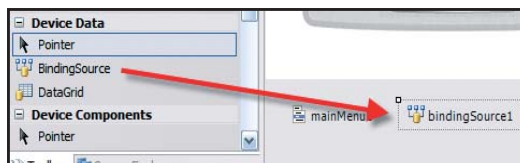



Fig. 8: ***Il BindingDataSource***

Il componente BindingSource viene trascinato nel relativo form e consentirà di associare i dati scelti ai controlli dell'interfaccia utente.

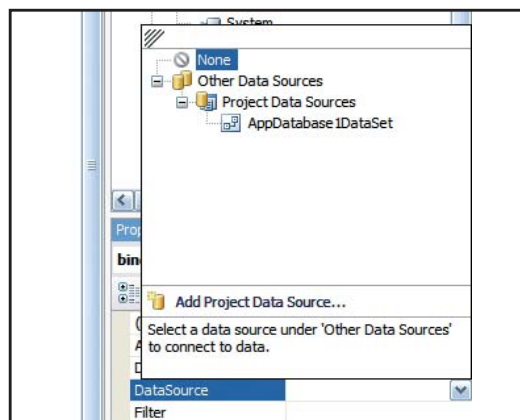



Fig. 9: ***Associamo la fonte dati***

Il secondo passo da fare, sarà quello di indicare al BindingSource, quale sarà la fonte da cui attingere i dati. Nel nostro caso, scegliamo il

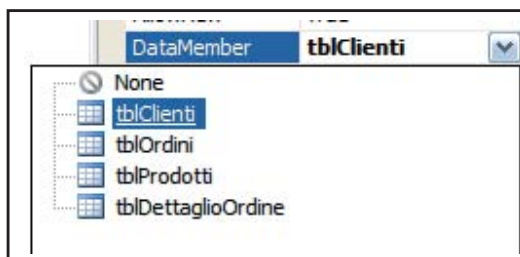



Fig. 10: ***Associamo le tabelle***

Data Set che abbiamo creato al momento della creazione del Data Base
L'ultimo step sarà quello di definire quale delle tabelle incluse nel DataSet, sarà associata al controllo. Nel nostro caso, selezioniamo la tabell tblClienti.
Da questo istante, possiamo far riferimento al nuovo BindingSource per creare associare i dati ai controlli. Nel prossimo paragrafo, analizzando la creazione del primo form, vedremo più in dettaglio come fare.

IL PRIMO FORM

Dopo aver visto come creare un Data Base per dispositivi mobili, come creare una fonte dati per i controlli Windows Forms e aver analizzato alcune linee guida per il posizionamento dei controlli, diamo realizziamo il primo form della nostra applicazione. Iniziamo quindi dalla scelta del cliente a cui associare l'ordine. Se stessimo sviluppando

Fig. 11: ***Il form con i dati dei clienti.***

una applicazione Windows, probabilmente basterebbe creare una DataGridView che mostri i clienti e dare la possibilità all'operatore di selezionarne uno direttamente dalla griglia. Sfortunatamente, con 250 pixel di larghezza, questa operazione potrebbe rivelarsi decisamente difficile. Ci conviene quindi strutturare il form con una griglia sintetica

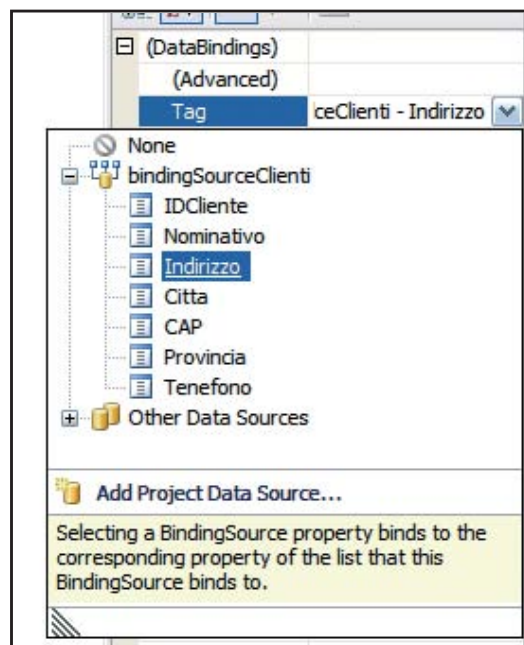



Fig. 12: ***L'associazione dei controlli alla fonte dati.***

un cui compare il nome del cliente e la città, e fornire i dettagli all'esterno della griglia, in delle normali label che andremo poi ad associare ai dati. Il form dovrebbe apparire come in figura.
Selezionando un cliente diverso dalla griglia in alto, i controlli in basso si aggiorneranno con i dati del cliente stesso. Questo comportamento è possibile in quanto sia la DataGrid che i controlli, hanno la stessa fonte dati che nel nostro caso è il BindingSource creato nel paragrafo precedente.
L'associazione dei controlli viene fatta, per ogni controllo, selezionando la voce (DataBinding) della finestra relativa alle proprietà. La stessa, comoda, tecnica è stata usata per tutti gli altri controlli. Questa combinazione di oggetti (Data Base, DataSet tipizzato e BindingSource) permette la creazione di applicazioni data oriented in tempi decisamente ridotti rispetto a quanto avveniva in passato.

LE FUNZIONALITÀ TELEFONICHE

Tra le novità di Windows Mobile 5.0, è stata sottolineato l'aumento di API che permettono di interagire con funzionalità dei Sistema Operativo direttamente dal mondo managed. Una di queste è la possibilità, per le versioni Phone Edition, di interagire con la parte telefonica in modo estremamente semplice. Nel form mostrato nel paragrafo precedente,

tra le informazioni relative al cliente c'è ovviamente il numero di telefono. Se volessimo chiamarlo per segnalargli il nostro ritardo, dovremmo appuntarci il numero da qualche parte, avviare l'interfaccia telefonica e comporre il numero. Grazie al namespace Microsoft.WindowsMobile. Telephony invece, questa operazione sarà possibile farla direttamente dall'interno dell'applicazione e con pochissimo codice:

```
private void btnChiama_Click(object sender,
                                    EventArgs e) {
    Phone myPhone = new Phone();
    myPhone.Talk(lblTelefono.Text, true);
}
```

Grazie a questo semplice codice, verrà avviata direttamente l'interfaccia telefonica e composto il numero, senza dover uscire dall'applicazione.

INTEGRAZIONE CON POCKET OUTLOOK

Altra funzionalità molto comoda è quella relativa all'integrazione con Pocket Outlook. Nella nostra applicazione ad esempio, potremmo far generare un appuntamento nel calendario coincidente con la consegna dell'ordine in modo che, al momento della consegna, il nostro utente può essere avvisato. In caso di gestione di più ordini, il rappresentante potrebbe anche organizzare al meglio gli appuntamenti intervenendo direttamente sul calendario.L'aggiunta di un appuntamento, grazie ancora alle nuove API esposte, è decisamente semplice:

```
Appointment appointment = new Appointment();
appointment.Subject = "Consegna ordine " +
                                   _CodiceOrdine;
appointment.Start = new DateTime(
    dateTimePicker1.Value.Year,
    dateTimePicker1.Value.Month,
    dateTimePicker1.Value.Day,
    10,
    00,
    00
    );
appointment.Duration = new TimeSpan(00, 30,
                                          00);
appointment.ReminderVibrate = true;
appointment.ReminderRepeat = true;

using (OutlookSession session = new
                              OutlookSession()){
  session.Appointments.Items.Add(appointment);
}
```

Fig. 13: ***Il nostro appuntamento inserito in PocketOutlook.***

Il risultato sarà quello di figura 6.

CONCLUSIONI

La nuova versione del Microsoft .NET Compact Framework, in aggiunta alle nuove API esposte da Windows Mobile 5.0, rende estremamente rapido lo sviluppo di applicazioni per dispositivi mobili, dando la possibilità a noi programmatori di creare applicazioni sempre più ricche e sempre più comode da usare. Ogni cosa però, va fatta sempre con moderazione. Usare i nuovi controlli come BindingSource anche quando non sono necessari, potrebbe rendere le applicazioni lente e scomode da usare, ottenendo di fatto il risultato opposto a quello per cui sono state create. Il compact framework è uno strumento veramente potente e flessibile. Probabilmente l'unico a consentire una modalità di sviluppo così rapida. Certamente la qualità dell'applicazione dipende dalle scelte effettuate in fase di design/progettazione, ma questa è una regola che vale sempre

Michele Locuratolo

L'AUTORE

Michele Locuratolo è Software Architect per la Mindbox S.r.l. di Capurso (BA), società che si occupa di consulenza e sviluppo software. E' cofondatore di DotNetSide.org, lo user group del Sud Italia il cui intento è quello di organizzare eventi di maggior spessore tecnico legati allo sviluppo con il .NET Framework. Il suo blog è raggiungibile all'indrizzo
http://www.dotnetside.org/blogs/mighell

E-COMMERCE FACILE IN JAVA

IN QUESTO ARTICOLO MOSTRIAMO COME UTILIZZARE UNO DEI SERVIZI DI POS ELETTRONICO PIÙ FAMOSI E UTILIZZATI IN ITALIA: EASYNOLO SVILUPPATO DA BANCA SELLA. SCOPRIREMO CHE È SUFFICIENTE UNA MANCIATA DI CODICE E QUALCHE API PER RENDERE TUTTO SEMPLICE

Quando vi recate in un negozio tradizionale ed effettuate un acquisto con la carta di credito, normalmente avvengono le seguenti operazioni:

- Fornite la vostra carta di credito alla gentile signora alla cassa
- Vi verrà chiesto un documento che certifichi la vostra identità
- La carta di credito verrà "inserita" in un POS, che invierà la richiesta di pagamento al circuito proprietario della carta di credito
- In qualche modo il circuito verificherà che la vostra carta disponga dei fondi necessari per effettuare il pagamento
- Se la carta è valida ed ha sufficientemente credito il pagamento verrà erogato, altrimenti verrà negato
- Se il pagamento è stato autorizzato verrà emesso uno scontrino sul quale apporrete una firma che autentica la vostra volontà di effettuare la transazione
- Il circuito proprietario della carta di credito si occuperà di trasportare il denaro dal vostro concorrente a quello dell'esercente

I soggetti che in un qualche modo partecipano a questa operazione sono realmente tanti. Normalmente il commerciante dispone di un POS fornitogli dalla propria banca. Il POS è pre-programmato per connettersi ai circuiti di verifica della carta e avviare la transazione. Ma nel caso del commercio elettronico, non esiste un POS pre-programmato in cui inserire la carta di credito. Quando si parla di e-commerce l'atto di inserire la carta nel POS sarà sostituito dall'inserimento dei dati relativi alla carta di credito all'interno di una Form. La banca non fornirà più all'esercente una macchina fisica in cui strisciare la carta, ma una serie di API/Servizi che dovranno essere richiamati all'atto della pressione del tasto "Invio" relativo alla form contenente i dati della carta di credito. GestPay è appunto un'insieme di API/Servizi resi disponibili da Banca Sella per la creazione di un POS virtuale.

CARATTERISTICHE DI GESTPAY

Il servizio di GestPay è diviso in due sezioni distinte. La prima sezione è costituita da una serie di componenti e di script specifici resi disponibile da Banca Sella, e da installare sul server che ospita il sito dell'esercente. La seconda sezione è costituita da un'area di BackOffice, ovvero un'area riservata messa a disposizione da Banca Sella sul proprio sito, alla quale l'esercente può accedere per visualizzare i report delle transazioni, effettuare storni o altre operazioni tipiche di un esercizio commerciale.
I componenti installati lato server espongono una serie di interfacce che il programmatore dovrà utilizzare per implementare il proprio servizio di POS. Saranno proprio questi componenti lato server che si occuperanno della trasmissione delle informazioni. Da un punto di vista strettamente tecnico, la trasmissione avverà su base SSL3 a 128 bit, per ottenere la massima sicurezza possibile.
GestPay permett di avere diverse tipologie di interfacce per il pagamento. Quella che vedremo in questo articolo è quella che viene denominata "Crittografia".

CRITTOGRAFIA

Questo metodo di pagamento segue un protocollo di sicurezza che si appoggia ad SSL. L'esercente reindirizza l'utente ad una pagina sicura di Banca Sella dove è possibile effettuare il pagamento. Se il pagamento viene eseguito in modo corretto, il controllo ritorna alle pagine del sito insieme ad un codice di risposta.
All'esercente non resta che verificare il codice di risposta, e dare il via alla catena di spedizione della merce se il pagamento risulta confermato, viceversa intraprendere altre azioni, se il pagamento non risulta confermato.
Uno schema di funzionanento del metodo di pagamento con crittografia è proposto in [[Figura 1]] L'utente inizia la comunicazione con il server dell'esercente. Molto probabilmente inserirà in un

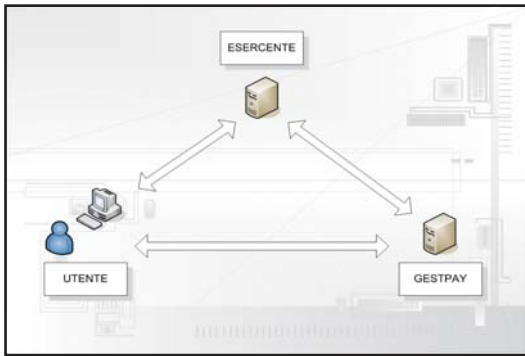



Fig. 1: ***L'architettura del sistema di pagamento con Crittografia***

carrello elettronico i prodotti che desidera comprare. Quando l'utente decide di effettuare il pagamento il server contatterà per la prima volta GestPay. Utilizzando una classe Java, che analizzeremo in dettaglio a brevissimo. Il server richiederà al componente Java di cifrare i dati della transazione e tenterà un collegamento al circuitoo GestPay. Quest'ultimo identificherà il nostro server e se lo riconoscerà autorizzato invierà una risposta. A questo punto potremo reindirizzare l'utente verso l'URL di pagamento di Banca Sella, passandogli come parametri quelli che ci sono stati forniti nella risposta da GestPay. L'utente effettua il pagamento inserendo il numero di carta di credito. Dopo di ciò GestPay comunica al nostro server e all'utente il risultato della transazione. Il nostro server viene contattato su una pagina dedicata, dove viene passato il risultato della transazione in maniera criptata. Questa informazione deve essere decriptata utilizzando di nuovo la classe Java di cui abbiamo già parlato. L'esito della transazione viene ricevuto anche dall'utente, il quale viene reindirizzato al nostro server. In questo modo abbiamo un doppio controllo sulla transazione che viene effettuata. Rispetto all'esito della transazione avremo due diverse pagine, una per l'esito positivo e l'altro per l'esito negativo. Tutte queste informazioni devono essere inserite nel pannello di BackOffice di GestPay.
Nel proseguio spiegheremo passo passo, proprio come realizzare questa serie di operazioni.

FORM DI PAGAMENTO

La prima cosa che dobbiamo fare è realizzare un form di pagamento, che ci consenta di effettuare l'interazione con il sistema di GestPay. In un sistema di produzione dovremmo anche inserire tutta una serie di informazioni riguardanti la sessione dell'utente, il numero di transazione, le informazioni sull'oggetto etc. etc. Per i nostri scopi didattici ci limiteremo a creare un semplice form html che richiamerà una pagina JSP che inizia la procedura di pagamento.

```
<pre>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
                                    Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type"
                  content="text/html; charset=UTF-8">
      <title>Form di pagamento</title>
   </head>
   <body>
   <h1>Form di pagamento</h1>
<form action="https://mioserver/pagamento.jsp">
<input name="myshoplogin" type="hidden"
                              value="GESPAY33374">
<input name="mycurrency" type="hidden"
                                       value="242">
Inserisci l'importo
<input type="text" name="importo" value="">
<br>
<input type="submit" value="Procedi con il
                       pagamento" name="submit">
</form>
</body>
</html>
</pre>
```

In questa pagina html non facciamo altro che inviare ad una pagina JSP dei parametri hidden che riguardano l'identificativo ell'utente (myshoplogin) e la valuta utilizzata (242 per l'euro). Inoltre viene inviato un parametro contenente la cifra da pagare per eseguire l'acquisto. Nel momento in cui testate questo codice con un vostro utente chiaramente inserite cifre piccole, che comunque vi verranno solamente sottratte dal platfond della vostra carta di credito e restituite successivamente.

RICHIESTA DI PAGAMENTO

Passiamo ora alla definizione della pagina di pagamento. Qui dovremo utilizzare l'oggetto GestPay CryptHS che viene reso disponibile per gli sviluppatori da EasyNolo. Praticamente passeremo a questo oggetto tutte le informazioni sulla nostra transazione e ci verranno restituiti i parametri per comporre l'url dove il nostro utente dovrà andare ad inserire le sue informazioni riservate.

```
<pre>
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="criptGP.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
```

```
                        Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type"
              content="text/html; charset=UTF-8">
    <title>Pagina di pagamento</title>
  </head>
  <body>
  <h1>Pagina di pagamento</h1>
  <%
  GestPayCryptHS objCrypt = new GestPayCryptHS();
  String myshoplogin =
              request.getParameter("myshoplogin");
  String mycurrency =
              request.getParameter("mycurrency");
  String myamount = request.getParameter("importo");
  String myshoptransactionID = "1";
  objCrypt.SetShopLogin(myshoplogin);
  objCrypt.SetCurrency(mycurrency);
  objCrypt.SetAmount(myamount);
 objCrypt.SetShopTransactionID(myshoptransactionID);
  //POSSIAMO ANCHE INSERIRE LE INFORMAZIONI
  // RIGUARDANTI L'UTENTE CHE ACQUISTA
  //QUESTE DEVONO ESSERE PREVENTIVAMENTE
                                      INSERITE
  // NEL BACKOFFICE DI GESTPAY
  //QUINDI IN QUESTO SEMPLICE CASO NON LE
  //UTILIZZIAMO MA LE LASCIAMO COMMENTATE
  /*
  String mybuyername = "Mario Rossi";
  String mybuyeremail = "mario.rossi@isp.it";
  String mylanguage = "1";
  String mycustominfo = "CODCLIENTE=1";
  objCrypt.SetBuyerName(mybuyername);
  objCrypt.SetBuyerEmail(mybuyeremail);
  objCrypt.SetLanguage(mylanguage);
  objCrypt.SetCustomInfo(mycustominfo);
  */
  objCrypt.Encrypt();
  String identificativo = "";
  String loginCifrato = "";
  if (objCrypt.GetErrorCode().equals("0")) {
    identificativo = objCrypt.GetEncryptedString();
    loginCifrato = objCrypt.GetShopLogin();
  }
  else {
    out.write ("GestPay Error");
    out.write ("GestPay ErrorCode
                    "+objCrypt.GetErrorCode()+"
                "+objCrypt.GetErrorDescription());
  }
  %>
  <form
   action="https://ecomm.sella.it/gestpay/pagam.asp">
    <input name="a" type="hidden"
                  value="<%=loginCifrato%>">
    <input name="b" type="hidden"
                  value="<%=identificativo%>">
    <input type="submit" value="Procedi con il
                  pagamento" name="submit">
  </form>
```

I TUOI APPUNTI

QUANTO COSTA GESTPAY

L'attivazione del servizio "Verified By Visa", che offre la maggiore sicurezza in relazioni a tentativi di frode ha un costo di 400 . Per ogni transazione è necessario poi corrispondere a Banca Sella il 3% sul valore dell'acquisto. Per i non correntisti di Banca Sella infine c'è un costo di 14.62 € dovuto come imposta di bollo. A tutto ciò è necessario aggiungere un canone per l'uso del software, da ricavare identificandolo sulla base della seguente tabella:

Caratteristiche	Basic	Advanced	Professional
Abilitazione a tutti i sistemi di pagamento	Si	Si	Si
Supporto valuta: Euro, Sterlina Inglese, Dollaro, Yen, Dollaro di Hong Kong, Real brasiliano.	No	Si	Si
Auto-Test	Si	Si	Si
Tool di gestione del rischio	No	Si	Si
Abilitazione a Verified by Visa	Si	Si	Si
Back-Office in lingua italiana, inglese e spagnola	Si	Si	Si
Back-Office con gestione multi-utente	Si	Si	Si
Personalizzazione pagina di pagamento	No	Si	Si
Active Report Light	Si	Si	Si
Active Report Full	No	No	Si
Modulo Server To Server	No	No	Si
Visibilità su Shop in Sella.it	Si	Si	Si
Campagna pubblicitaria e banner su Shop in Sella.it	No	No	Si
Costo Mensile	7,75 + iva	12,91 + Iva	18,08 + Iva

```
    </body>
</html>
</pre>
```

Come potete vedere il codice è abbastanza semplice e intuitivo. Dopo che l'utente ha cliccato sul bottone della form html la nostra applicazione rimane in attesa, ovvero noi usciamo dal dialogo con l'utente, il quale parla direttamente con GestPay.

L'AMBIENTE DI BACKOFFICE

Abbiamo già detto che quando GestPay riceve la richiesta di connessione deve identificare il server richiedente come valido. Per tale motivo l'esercente deve configurare nel proprio ambiente di Back-Office l'indirizzo IP del server che effettuerà le richieste.

Fig. 2: Configurazione IP

Allo stesso modo devono essere segnalate le pagine che devono ricevere l'esito della risposta. È necessario configurare sia una pagina per esiti positivi sia una pagina per esiti positivi.

Fig. 3: Configurazione pagine di risposta

Le pagine di risposta sarsanno presenti sul server dell'esercente e dovranno contenere il codice necessario a gestire la transazione sia in caso di risposta positiva sia in caso di risposta negativa

PAGINA DI PAGAMENTO

Dopo aver cliccato sul bottone della form html l'utente visualizza con il browser il pagamento online di BancaSella. Questo pagamento prevede la possibilità di utilizzare carte di credito differenti e ticket prepagati. Dopo aver inserito i dati vengono chieste le classiche conferme e infine si viene ridirezionati sul nostro server.

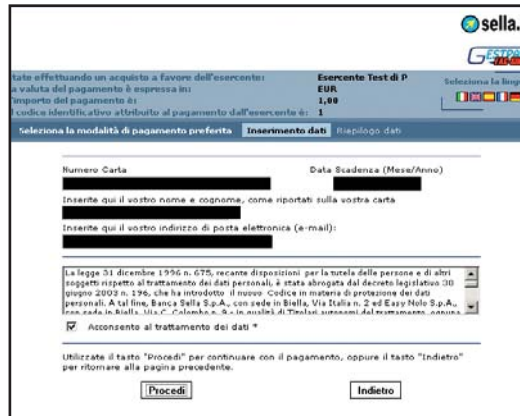



Fig. 4: Il pagamento che l'utente visualizza con il browser

PAGINA DI RISPOSTA

In questo esempio utilizziamo una sola pagina di risposta, la quale scriverà semplicemente in output tutti i dati restituiti dalla transazione. Anche in questo caso utilizziamo la classe GestPayCryptHS per decripta i dati che ci vengono inviati.

```
<pre>
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="criptGP.*"%>
<html>
    <head>
        <meta http-equiv="Content-Type"
                    content="text/html; charset=UTF-8">
        <title>RISPOSTA DEL PAGAMENTO
</title>
    </head>
    <body>
    <h1>Reply Page</h1>
    PAGINA PER LA GESTIONE DELLA RISPOSTA DI
                                        PAGAMENTO
    <%
    String
shoptrxID,buyername,buyeremail,trxresult,authcode
errorcode,errordescription,banktrxid,alertcode,alert
                                escription,custominfo;
    String parametro_a = request.getParameter("a");
    String parametro_b = request.getParameter("b");
    parametro_a = parametro_a.trim();
    parametro_b = parametro_b.trim();
```

```
    GestPayCryptHS objdeCrypt = new
                                GestPayCryptHS();
objdeCrypt.SetShopLogin(parametro_a);
objdeCrypt.SetEncryptedString(parametro_b);
objdeCrypt.Decrypt();
String ErrorCode=objCrypt.GetErrorCode();
String shoplogin =
                objdeCrypt.GetShopLogin().trim();
int currency =
      Integer.parseInt(objdeCrypt.GetCurrency());
Float amount = new
                Float(objdeCrypt.GetAmount());
String shoptrxID =
        objdeCrypt.GetShopTransactionID().trim();
String buyername =
              objdeCrypt.GetBuyerName().trim();
String buyeremail =
              objdeCrypt.GetBuyerEmail().trim();
String trxresult =
        objdeCrypt.GetTransactionResult().trim();
String authcode =
          objdeCrypt.GetAuthorizationCode();
String errorcode = objdeCrypt.GetErrorCode();
String errordescription =
       objdeCrypt.GetErrorDescription().trim();
String banktrxid =
      objdeCrypt.GetBankTransactionID().trim();
String alertcode =
              objdeCrypt.GetAlertCode().trim();
String alertdescription =
       objdeCrypt.GetAlertDescription().trim();
String custominfo =
             objdeCrypt.GetCustomInfo().trim();
out.write("ID di transazione del negozio
                          "+shoptrxID+"<br>");
out.write("Importo "+amount+"<br>");
out.write("Nome del compratore
                          "+buyername+"<br>");
out.write("Email del compratore
                          "+buyeremail+"<br>");
out.write("Risultato della transazione
                          "+trxresult+"<br>");
out.write("Codice di autorizzazione
                          "+authcode+"<br>");
out.write("Codice d'errore "+errorcode+"<br>");
out.write("Descrizione dell'errore
                   "+errordescription+"<br>");
out.write("ID di transazione della banca
                          "+banktrxid+"<br>");
out.write("Codice di Alert "+alertcode+"<br>");
out.write("Descrizione dell'Alert
                   "+alertdescription+"<br>");
out.write("Custom info "+custominfo+"<br>");
System.out.println("RICHIESTA DA
                   "+request.getRemoteHost());
%>
</body>
</html>
</pre>
```

Come potete vedere utilizziamo anche in questo caso un package criptGP. Questo è stato definito per questo esempio, inserendo il codice sorgente che viene fornito della classe GestPayCryptHS. Questo viene fatto soltanto per poter distribuire l'applicazione con un solo pacchetto war e per poter definire un nostro package che possiamo ampliare con ulteriori classi riguardanti la sicurezza del nostro sito. Insieme alla visualizzazione di questa pagina abbiamo come risultato finale anche l'invio di email all'utente e all'amministratore del sito per la segnalazione della transazione. Nell'ambiente di test che viene fornito possiamo cancellare la transazione utilizzando il menu Active Report -> Report. Oltre alla configurazione del sistema, nell'ambiente di BackOffice troviamo anche una reportistica per il nostro sito di ecommerce, utile per controllare la situazione generale.

OTP

GestPay fornisce anche un altro metodo di pagamento differente da "Crittografia". Questo si basa sulla generazione di OTP (One Time Password). Praticamente dal BackOffice viene richiesta la generazione di un certo numero di password. Quando la generazione è completata ci viene segnalato tramite email e possiamo procedere all'attivazione e il download di queste password. Di volta in volta utilizzeremo queste password nel nostro sito, considerando che valgono per una sola transazione e che una volta finite dovremo provvedere manualmente all'aggiornamento.

CONCLUSIONI

Il sistema di pagamento che viene fornito da Banca Sella permette di costruire il modulo di pagamento del nostro sito di e-commerce in una maniera abbastanza semplice. In questo modo infatti possiamo agevolmente creare un modulo di pagamento che viene integrato nel nostro sistema, fornendo all'utente un pratico pagamento con carta di credito. In Italia quasi tutti le banche forniscono sistemi di pagamento virtuale. Il metodo generale è quasi sempre molto simile a quello proposto in questo articolo, a cambiare sono ovviamente le interfacce esposte dai vari componenti. In ogni caso se avete un cliente che desidera aprire un negozio online è sempre opportuno chiedere alla propria banca quali sono i servizi offerti. Molto spesso si ottengono condizioni personalizzate adeguate ai propri volumi di traffico

Federico Paparoni

CLASSI AL VOLO CON IL CODICE DINAMICO

LA VERSIONE 2.0 DI .NET È TALMENTE PERSONALIZZABILE CHE L'INTERO COMPILATORE È ESPOSTO IN CLASSI. QUESTA CARATTERISTICA PUÒ ESSERE SFRUTTATA PER FAR GESTIRE A CLASSI SPECIALI FILE PARTICOLARI...

Diciamo subito che parlare di Build Providers non è esattamente cosa semplice. Allora prima di entrare nel merito della teoria, è il caso di iniziare con un sempio che ci chiarirà meglio le idee su quello che ci attende. I passi che intendiamo eseguire sono i seguenti:

- Disponiamo di uno o più file XML rinominati con estensione .iop e contenuti nella directory "APP_CODE" di un progetto ASP.NET
- Un template per i file in questione potrebbe essere il seguente:

```
<greetings title="greet1">Hello World</greetings>
```

- desideriamo che quando l'applicazione viene lanciata, tutti i file contenuti nella directory APP_CODE e con estensione .iop vengano parserizzati.
- Per ogni file parserizzato deve essere generata "dinamicamente" una classe che possiamo instanziare all'interno della Web Application.

Questo ci consente di non dover creare fisicamente i file delle classi all'interno del progetto, ma di poterle instanziare dinamicamente. Come fare?
Iniziamo con il creare una nuova soluzione dotata di un progetto "Class Library" in C#. Aggiungiamo un riferimento a "System.Web" e scriviamo la nostra classe come segue:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Web;
using System.Web.Compilation;
using System.CodeDom;
using System.Xml;
using System.IO;
namespace iopBuildProvider
{
    public class iopBuilder: BuildProvider
    {
        private XmlDocument IOPDocument;
        private XmlDocument LoadDocument()
        {
            if (this.IOPDocument == null)
            {
                using (Stream stream = this.OpenStream())
                {
                    XmlDocument iopDocument = new
                                        XmlDocument();
                    iopDocument.Load(stream);
                    IOPDocument = iopDocument;
                }
            }
            return this.IOPDocument;
        }
        private string Title
        {
            get
            {
                XmlDocument iopDocument =
                                this.LoadDocument();
                XmlNode n =
                iopDocument.SelectSingleNode("/greetings");
                return n.Attributes["title"].Value;
            }
        }
        public override void GenerateCode(AssemblyBuilder
                                        assemblyBuilder)
        {
            CodeTypeDeclaration myType = new
                                CodeTypeDeclaration(Title);
            CodeMemberMethod myMethod = new
                                CodeMemberMethod();
            myMethod.Name = MyMethod;
            myMethod.ReturnType = new
                        CodeTypeReference(typeof(String));
        }
    }
}
```

Aggiungiamo un Web Site al progetto, e modifichiamo il web.config aggiungendo le seguenti righe:

```
<compilation debug="true" defaultLanguage="C#">
<buildProviders>
```

REQUISITI

Conoscenze richieste
HTML, ASP.NET 2.0

Software
Microsoft .NET Framework 2.0, ASP.NET 2.0

Impegno

Tempo di realizzazione

```
<add extension=".iop" type="iopBuildProvider.iopBuilder,
                                        iopBuildProvider"/>
</buildProviders>
</compilation>
Nela directory APP_CODE aggiungiamo un file
                                   progetto.iop contenente:
<greetings title="greet1">Hello World</greetings>
Instanziamo la nuova classe come segue:
<script runat="server">
void Page_Load(object sender, EventArgs e)
{
greet1 g = new greet1();
}
</script>
```

Cosa abbiamo fatto? Modificando il web.config abbiamo fatto in modo che tutti i file con estensione iop presenti in APP_CODE vengano gestiti dalla libreria ioPbuilder. ioPbuilder contiene un Build Provider che attraverso il metodo *GenerateCode* compila in C# la nuova classe e la rende disponibile all'applicazione. Ovviamente per ottenere la classe in VB sarebbe sufficiente semplicemente cambiare il parametro nel web.config.

DALLA PRATICA ALLA TEORIA

Dovrebbe essere chiaro che un build provider è essenzialmente questo: un generatore e compilatore di codice. D'altra parte il nome è abbastanza esplicativo e ci aiuta anche nel restringerne l'area di utilizzo. Un build provider, infatti, è legato ad un particolare tipo di risorsa ed anzi agisce proprio in base all'estensione. Ecco perché un file con estensione .aspx ci restituisce una pagina, o quelli con estensione .Master ci consentono di avere una classe che rappresenti una Master Page: sono i build providers, in maniera del tutto automatica, a farlo per noi, proprio come nel caso della compilazione delle pagine (del cui ciclo in realtà fanno parte).
Non è certo argomento semplice da padroneggiare, ma i concetti sono semplici: ci sono delle classi che "leggono" il contenuto di alcuni file e "traducono" questo contenuto in codice, che poi viene compilato, come qualsiasi altra cosa all'interno di ASP.NET. Prendiamo come esempio una pagina basata sul CodeFile (cioè con il codice separato dal markup). Sarà composta da un primo file, qualcosa così:

```
<%@ Page language="C#" CodeFile="Test.aspx.cs"
                                        Inherits="Test" %>
Questa è una <asp:Label runat="server" id="label1"
                                         Text="label" />
```

Ed un secondo che invece è il codice server-side vero e proprio:

```
public partial class Test : System.Web.UI.Page {
    protected void Page_Load(object sender,
                                        EventArgs e)
    {
        label1.Text="label";
    }
}
```

In realtà quando richiamiamo la nostra pagina test.aspx, ASP.NET produrrà una classe che avrà all'incirca questo contenuto, dal nome app_web*.cs, dove * indica una variante univoca all'interno dell'applicazione, per evitare conflitti:

```
public partial class Test : {
    protected
          global::System.Web.UI.WebControls.Label label1;
}
public class test_aspx : global::Test {
    private global::System.Web.UI.WebControls.Label
                            __BuildControllabel1() { ... }
    private void __BuildControlTree(test_aspx __ctrl)
                                              { ... }
}
```

In pratica verranno unite le due classi, laddove il markup, da insieme di testo HTML e dichiarazione di controlli, diventa invece una serie di definizioni ed istanze di controlli, con tanto di proprietà impostate secondo il nostro codice originale, che poi sarà compilato sfruttando il linguaggio con in quale abbiamo deciso di scrivere la nostra applicazione.

COME FUNZIONA UN BUILD PROVIDER

Un build provider internamente non fa altro che generare codice, come abbiamo ormai capito. In pratica, spesso un build provider ha un proprio sistema di analisi del codice sorgente, che provvede

METTERE I FILE IN QUALSIASI DIRECTORY

Il manager dei build providers crea un assembly per ogni directory, quindi le classi dei nostri file .aspx (o di qualsiasi altro tipo) finiscono tutte insieme e si vedono tra di loro, come se fossero normalissimi oggetti creati "manualmente". Se per questo si vuole utilizzare per esempio il nostro build provider di file Excel posizionandoli in un'altra directory, allora bisogna referenziare il percorso nella pagina.
Per farlo basta aggiungere questa direttiva alla pagina ASP.NET:

```
<%@ Reference
     virtualPath="cartella/file.xls" %>
```

In questo modo viene aggiunto il riferimento all'assembly corrispondente creato attraverso il nostro build provider. Questo problema non c'è per la directory /app_code/ perché è automaticamente referenziata da ogni risorsa.

a tradurre poi in qualcosa che il .NET Framework, su cui si base ASP.NET, possa interpretare. Una caratteristica del .NET Framework è infatti proprio quella di essere indipendente dal linguaggio, grazie al fatto che in realtà si basa su IL (Intermediate Language). E proprio per preservare questa caratterista, si sfrutta quello che in gergo è chiamato CodeDOM. Come il nome suggerisce, analogamente a XMLDOM o Javascript DOM, è un insieme di funzionalità che servono a gestire codice.
Una di queste consente funzionalità di generarlo, per l'appunto, indipendentemente dal linguaggio prescelto. Torniamo ancora una volta all'esempio della nostra pagina. Come sappiamo, è possibile scriverle in uno qualsiasi dei linguaggi supportati dal .NET Framework. Insomma spesso una pagina è scritta in C#, ma tante altre volte lo è in VB 2005. Ed un build provider deve essere egualmente capace di generare codice, altrimenti poi la classe che viene creata sarebbe composta da codice scritto in linguaggi diversi, e pertanto non sarebbe compilabile. È proprio in questo scenario che CodeDOM garantisce una generazione del codice indipendente dal linguaggio della nostra applicazione.

NOTE

INFORMAZIONI SU CODEDOM

Potete trovare ulteriore documentazione su CodeDOM su MSDN, a questi indirizzi:
http://msdn.microsoft.com/library/enus/cpgenref/html/cpconCodeDOMQuickReference.asp
http://msdn.microsoft.com/library/enus/cpgenref/html/cpconCodeDOMQuickReference.asp

GENERARE CODICE: IL CODEDOM

Dunque CodeDOM risponde a questa necessità, consentendoci di concentrarci sulle funzioni di generazione del codice, più che sul codice derivante dal risultato finale.
Il concetto è che dobbiamo scrivere codice per generare codice, secondo la logica del nostro build provider.
Per esempio, questo è il codice in C# necessario per generare una classe di nome MyClass con un metodo che restituisce una stringa, di nome MyMethod:

```
CodeTypeDeclaration myType = new
                    CodeTypeDeclaration("MyClass");
CodeMemberMethod myMethod = new
                    CodeMemberMethod();
myMethod.Name = "MyMethod";
myMethod.ReturnType = new
                    CodeTypeReference(typeof(String));
```

E questo è il relativo codice che sarà generato in C#:

```
public class MyClass
{
public String MyMethod()
{
}
}
```

Ma dalla stesso codice di partenza, scritto in C#, avremo questa classe in VB 2005:

```
Public Class MyClass
Public Function MyMethod() As String
End Function
End Class
```

Il CodeDOM è implementato attraverso una serie di oggetti che si trovano nel namespace System.CodeDom. Il modello ad oggetti fornito ci consente praticamente di "descrivere" una classe in modo ricco e senza rinunciare a nessuna delle caratteristiche che avremmo scrivendo direttamente il codice.
Come già detto è un modo di generare codice indipendente dal linguaggio, che si basa quindi su quelle che sono le caratteristiche di IL e che soprattutto sfrutta generatori di codice già pronti, basati sull'interfaccia ICodeGenerator.
Il vantaggio di CodeDOM è che si può generare codice in qualsiasi linguaggio supporti le specifiche, si può compilarlo ed eseguirlo, tutto a runtime, senza avere fisicamente un oggetto disponibile su disco. Nel namespace troviamo il supporto per quasi tutti i costrutti di programmazione: dichiarazioni, istruzioni, cicli, conversioni, array, commenti, gestione degli errori. Ma ci sono anche delle mancanze, che però possono essere superate utilizzando le classi Snippet.
Il concetto è simile ai controlli literal di ASP.NET: in pratica, possiamo specificare del codice da compilare direttamente. Esistono diversi tipi di snippet: CodeSnippetCompileUnit, CodeSnippet Expression, CodeSnippetStatement e CodeSnippetTypeMember, questo per rendere possibile ogni tipologia di scenario.
Tuttavia la limitazioni che ci sono non rappresentano grosse problematiche, perché sono comunque facilmente aggirabili. Non è ad esempio possibile dichiarare l'alias di un namespace, che non possono nemmeno essere annidati, così come non è possibile dichiarare variabili di uno stesso tipo tutte insieme e non è possibile utilizzare il modifier "unsafe" in C#, per dichiarare in questo modo un metodo.
Per meglio orientarvi in quello che potete fare, nella tabella a fianco è incluso l'elenco completo delle classi con relativa spiegazione.
Oltre dai build provider, CodeDOM è utilizzato da tool come WSDL.exe o XSD.exe, che se avete mai usato, saprete certamente che hanno la caratteristica di generare codice (nel linguaggio che preferiamo) partendo da uno schema, rispettivamente un Web Service o uno schema XML.
Difatti il motivo per cui uno potrebbe aver bisogno di un build provider personalizzato è proprio quello di utilizzare un proprio formato di definizione ed avere invece accesso in maniera tipizzata agli oggetti risultanti.

CREARE UN BUILD PROVIDER

I tipici esempi sono quello di definire classi attraverso una sintassi XML, creare dei ORM (Object Relational Mapper) semplificati, dove si ha una corrispondenza tra modello ad oggetti risultante e struttura del database. Piuttosto che accedere in maniera tipizzata alle proprietà del web.config o al contenuto di un file in formato Excel, o a qualsiasi altra cosa vi passi per la testa. Il cuore di tutto, in ogni caso, sarà proprio il Build Provider, che con l'uso del CodeDOM andrà a generare il codice.
Per il nostro esempio ci baseremo su un build provider creato da Cristian Civera, disponibile anche nel cd allegato alla rivista, facilmente estendibile, che ha il vantaggio di darci accesso in maniera del tutto strongly typed a qualsiasi sheet di qualsiasi documento Excel, semplicemente copiandolo nella directory /App_Code/, come potete notare in questo screenshot, con l'Intellisense di VS 2005 che è in grado di farci accedere allo stesso come se stessimo usando un oggetto qualsiasi:

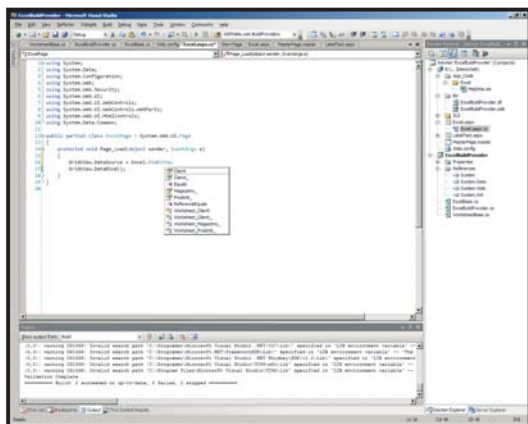



fig.1 si noti il funzionamento dell'intellisense

Come si può notare viene creato al volo un namespace Excel, all'interno del quale saranno costruiti in fase di esecuzione, sfruttando CodeDOM, le istanze di tutti i file e dei relativi worksheet, così da poter accedere in maniera strongly typed a questi ultimi. Analizzando il metodo GenerateCatalog della classe ExcelBuildProvider si potrà notare come viene utilizzato il CodeDOM in scenari reali: ci sono molte chiamate ed il codice è tutt'altro che banale, ma il risultato è ovviamente quello di creare, a tutti gli effetti, classi al volo, leggendone la struttura da sorgenti esterni, nel nostro caso un foglio Excel. Il tutto è fatto derivando dalla classe base BuildProvider, da cui tutti i build provider ovviamente derivano. Se andiamo a dare un'occhiata al contenuto della \v2.0.50727\Temporary ASP.NET Files\, scegliendo la directory con il nome della nostra applicazione, noteremo una directory di nome Sources_App _Code all'interno della quale sono stati savlati i file autogenerati dal nostro BuildProvider (ma è necessario tenere in debug l'applicazione perché siano presenti i sorgenti). E se facciamo attenzione, noteremo che all'interno del file c'è proprio uno dei commenti che abbiamo scritto sfruttando la classe CodeCommentStatement!

UTILIZZARE UN BUILDPROVIDER

La parte più semplice è davvero quella di utilizzare il build provider. Perché ASP.NET (e VS 2005) possa utilizzarlo, è sufficiente provvedere alla registrazione, sfruttando la sezione \configuration\ system.web\compilation\ buildProvider, in questo modo:

```
<configuration>
        <system.web>
        <compilation defaultLanguage="C#">
        <buildProviders>
          <add extension=".xls"
                type="ASPItalia.com.BuildProviders.
             ExcelBuildProvider, ExcelBuildProvider"/>
        </buildProviders>
        </compilation>
        </system.web>
</configuration>
```

Da questo momento in poi, ogni file .xls (estensione specificata attraverso la proprietà extension) invocherà la classe ExcelBuild.Provider, che essendo quella che abbiamo appena creato, provvederà prima a creare attraverso il CodeDOM il codice nel linguaggio (nel nostro caso C#, ma provate a cambiare l'attributo defaultLanguage del tag compilation e notate la differenza…) e poi a compilarlo in un oggetto, da utilizzare all'interno delle nostre applicazioni. Ci basta aggiungere un file Excel per avere accesso nello stesso modo, senza fare nient'altro, alle relative informazioni. Provare per credere!

CONCLUSIONI

L'architettura di HttpRuntime in ASP.NET 2.0 è profondamente differente e consente molta più estendibilità rispetto alla versione 1.1. I build provider ne sono un ottimo esempio e consentono di costruire funzionalità sfruttando strutture più semplici, come file XML, o anche più complesse, come database. Ancora una volta, l'unico limite è nella nostra fantasia!

Daniele Bochicchio

Daniele Bochicchio è il content manager di ASPItalia.com, community che si occupa di ASP.NET, Classic ASP e Windows Server System. Il suo lavoro è principalmente di consulenza e formazione, specie su ASP.NET, e scrive per diverse riviste e siti. E' Microsoft ASP.NET MVP, un riconoscimento per il suo impegno a supporto delle community e per l'esperienza maturata negli anni. Il suo blog è all'indirizzo http://blogs.aspitalia.com/daniele/

LOGIN PROTETTO CON IMMAGINI CAPTCHA

VI È MAI CAPITATO DI DOVER RIPORTARE IL TESTO RAPPRESENTATO IN UN'IMMAGINE ALL'INTERNO DI UNA FORM, PER POTER EFFETTUARE LA REGISTRAZIONE A UN SITO WEB? ECCO COME POTETE IMPLEMENTARE QUESTA FUNZIONE NEI VOSTRI SITI

La maggioranza dei siti Web offrono la possibilità di registrarsi, al fine di ottenere un'identificazione per usufruire dei servizi esposti: forum, wiki, blog, etc. I dati di registrazione vengono forniti attraverso form Web. Recentemente però gli spammer hanno creato dei bot che scansionano la rete alla caccia di pagine di registrazione, e quando ne trovano eseguono una registrazione fasulla, con fini non proprio leciti. Per proteggersi da questo tipo di attacco, un buon metodo è quello di usare "immagini captcha". Si tratta di un'immagine generata in maniera random e contenente una stringa o un codice. L'unico modo per validare la registrazione è quello di riportare il contenuto dell'immagine in un campo di confronto, operazione che può essere fatta solo manualmente dall'utente. Difficilmente un bot sarebbe in grado di analizzare il contenuto dell'immagine e riempire il campo di confronto.

Fig. 1: La pagina di registrazione di un nuovo account in Yahoo e l'immagine con il testo "distorto"

Nel 2000 alcuni autori, della Carnegie Mellon University, hanno proposto il termine CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) per designare tutti quei metodi che prevedono la generazione random di un oggetto e la conseguente validazione basata sulla sull'interpretazione dell'oggetto (infatti si potrebbe pensare a metodi alternativi alla generazione di immagini quali suoni o altre forme di comunicazione).

Lo scopo dell'articolo è generare immagini CAPTCHA e renderle disponibili attraverso un Web Service scritto in Java. Per farlo si useranno alcuni progetti open source (in particolare JCaptcha, Axis) e si mostrerà come utilizzare il servizio in alcune pagine Web.

INSTALLARE ED USARE JCAPTCHA

Esistono molti progetti Open Source nati con l'intento di realizzare framework per la generazione di immagini CAPTCHA. In ambiente Java uno dei progetti più interessanti è JCaptcha (sito di riferimento http://www.jcaptcha.net).

Per prima cosa è necessario eseguire il download del progetto (va bene il progetto in forma binaria, ovvero già compilato: jcaptcha-bin-1.0-RC2.0.1.zip). Per installarolo è sufficiente prendere il file jar contenuto nell'archivio compresso e inserirlo nel classpath prima di compilare il proprio progetto. Il nostro esempio è stato sviluppatp usando Eclipse, ma va bene un qualsiasi editor, IDE o semplice compilatore!. Va anche fatto il download del file ehcache-1.2.jar dal sito http://ehcache.sourceforge.net/ (esso viene utilizzato internamente da JCaptcha) e posto nel classpath del progetto.

GENERARE LE IMMAGINI

La classe *com.octo.captcha.service.image.DefaultManageableImageCaptchaService* permette di reperire un servizio per la creazione di nuove immagini (servizio con parametri di default; è anche possibile creare servizi personalizzati impostando le caratteristiche di creazione delle immagini, come font da usare, colori, alfabeto per i caratteri e così via). È importante accedere ad un'unica istanza del servizio, ovvero usare un pattern come il Singleton. Ecco un esempio di classe che realizza un singleton e che restituisce il servizio di default:

```
public class ServiceSingleton {
  private static ImageCaptchaService instance =
     new DefaultManageableImageCaptchaService();

  public static ImageCaptchaService getInstance() {
    return instance;
  }
}
```

Tale servizio ha un metodo (*getImageChallengeForID*) che permette di generare una nuova immagine e di associarla ad un identificativo univoco. Ecco, per esempio, come sfruttare il servizio per generare una nuova immagine (l'identificativo, in questo caso, viene passato come parametro):

```
public static byte[] getImage(String captchaId) {
  ByteArrayOutputStream jpegOutputStream = new
     ByteArrayOutputStream();
  try {
    BufferedImage challenge = ServiceSingleton
      .getInstance()
      .getImageChallengeForID(captchaId);
    JPEGImageEncoder jpegEncoder = JPEGCodec
      .createJPEGEncoder(jpegOutputStream);
    jpegEncoder.encode(challenge);
  } catch (Exception e) {
    return null;
  }
  return jpegOutputStream.toByteArray();
}
```

Il metodo si occupa anche di generare un array di byte che rappresentano l'immagine secondo la codifica JPEG. Dallo stesso servizio è possibile verificare se una risposta è corretta per l'immagine usando il metodo *validateResponseForID* (in questo caso i parametri sono due: l'identificativo con cui è stata generata l'immagine e la risposta inserita dall'utente; quest'ultima viene memorizzata come stringa di caratteri):

```
public static boolean isCorrect(String id,
                                String tryResponse){
  return
  ServiceSingleton.getInstance().validateResponseForI
                                  D(id, tryResponse);
}
```

Questi due metodi sono, nell'esempio proposto, inseriti nella classe *ImageUtility* e, da soli, sono sufficienti a “comandare” JCaptcha per la generazione delle immagini e il test sul valore inserito. La classe *ImageUtility* ha anche un metodo che permette di generare un nuovo identificativo univoco: *getNextId*. Come parametri accetta due stringhe che rappresentano, rispettivamente, uno username ed una password. Se essi sono tra le coppie ritenute valide allora viene generato l'ID, altrimenti viene restituito null (per i dettagli implementativi si faccia riferimento ai sorgenti della classe).

CREARE IL WEB SERVICE

Per creare il Web Service si utilizzerà Axis, nella versione 1.3. Dopo aver scaricato la versione binaria del framework e averla scompattata, si prenda la cartella “axis” presente sotto “webapps” e la si copi nella cartella $CATALINA_HOME/webapps (dove per $CATALINA_HOME di intende la cartella dov'è installato Tomcat).
Esiste un modo molto semplice per la generazione di un servizio: è sufficiente creare una classe Java che non appartenga ad alcun package e rinominare il file .java in .jws, quindi posizionarlo nella cartella della WebApp di Axis. Essa verrà compilata e resa disponibile come Web Service (affinché venga compilata è necessario che Tomcat contenga nel path il file tools.jar ed eventuali classi/archivi di cui la classe fa uso). Se si vuole compilarla esternamente, basterà mettere il file .class nella cartella axis/WEB-INF/jwsClasses della WebApp Axis (anche in questo caso va comunque inserito il file .jws; semplicemente quest'ultimo non verrà compilato più da Tomcat).

```
public class CaptchaWebService {
  public static String getNextId(String user, String
                                    mySecretWord){
    return ImageUtility.getNextId(user,
                                    mySecretWord);
  }
  public byte[] getImage(String id) {
    return ImageUtility.getImage(id);
  }
  public boolean isCorrect(String id, String
                                    tryResponse){
    return ImageUtility.isCorrect(id, tryResponse);
  }
}
```

Ovviamente la classe ImageUtility, insieme ad altre classi usate, vanno copiate nella cartella axis/WEB-INF/classes, mentre i JAR (in questo caso quelli di JCaptcha) vanno posti in axis/WEB-INF/lib.
Come si può osservare la classe non fa altro che esporre i metodi presenti nella classe *ImageUtility*. Perché, allora, non esporre direttamente quest'ultima? Innanzi tutto essa appartiene ad un package (*it.ioprogrammo.captcha*): eliminarlo potrebbe essere una soluzione per questo caso, ma così facendo non si potrebbe usare la classe nelle JSP (per esempio). Insomma: molto meglio creare una nuova classe che è un semplice “wrapper” e lasciare disaccoppiata la realizzazione vera e proprio delle funzionalità.

Per eseguire il test del servizio di può digitare l'url a cui risponde (data dal nome, o indirizzo ip, dell'host, la porta a cui risponde Tomcat, il nome della webapp di Axis più il nome del file .jws; per esempio: http://127.0.0.1:8080/axis/CaptchaWebService.jws). Se tutto è andato a buon fine appare una pagina semplice semplice, con un link per ottenere il WSDL ("Click to see the WSDL", Figura 2). Il WSDL è, a tutti gli effetti, un descrittore che permette a chiunque (e in qualunque linguaggio!) di generare dei client che facciano uso del Web Service appena creato.

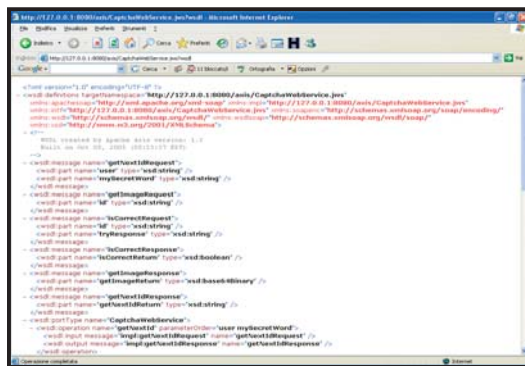



Fig. 3: ***Il WSDL generato***

CREARE LE CLASSI PER ACCEDERE AL WS

È possibile creare un client usando ancora Axis, ma è possibile farlo usando anche uno dei linguaggi della piattaforma .NET o PHP, Python, Perl o qualsivoglia altro linguaggio. Anche in questo caso i framework aiutano notevolmente la scrittura delle classi. Axis, per esempio, permette di generare in automatico tutte le classi che "incapsulano" la logica di interazione con il Web Service ed espongono il servizio remoto come chiamate a metodi locali. Per farlo è necessario settare il classpath in maniera che contenga tutte le lib del progetto, salvare il file WSDL e poi invocare il comando:

```
java org.apache.axis.wsdl.WSDL2Java
                              CaptchaWebService.wsdl
```

Provando a generare le classi si nota che esse appartengono al package _1._0._0._127; questo può risultare sorprendente, ma è dovuto al fatto che il WSDL è stato scaricato da una url che conteneva l'ip della macchina locale. Per ovviare a questo problema si può eseguire lo stesso comando precedente ma con una nuova opzione:-p it.ioprogrammo.axis (nome del package).
Benché sia semplice utilizzare anche la generazione usando direttamente le classi di Axis, esistono numerosi tool che integrano l'intero sviluppo all'interno dei più diffusi IDE; Eclipse possiede il plug-in WTP (Web Tools Project) che, tra le altre cose, gestisce la creazione di Web Services (sia lato server che lato client) usando Axis come framework.

UNA WEBAPP COME CLIENT

Non resta che usare le classi di accesso al WS in una applicazione. Per esempio si potrebbe creare un nuovo progetto web che mostri l'immagine e, tramite una form, chieda all'utente di inserire il testo che legge in essa:

```
<form name="my" action="risultato.jsp"
  method="post">
<%
  CaptchaWebServiceServiceLocator loc =
     new CaptchaWebServiceServiceLocator();
  CaptchaWebService_PortType ws =
     loc.getCaptchaWebService();
  String id = ws.getNextId("user2", "qwertyuiop");
%>
  <input type="hidden" name="id_ws"
     value="<%= id %>" />
  <input type="text" name="testo_random" />
  <input type="submit" name="Avanti" />
</form>
```

Il primo problema da affrontare è quello di decidere come l'immagine debba essere memorizzata; infatti essa dovrebbe venire creata dal WS, ma come la referenzia la pagina HTML? Ovviamente ci sono più possibili soluzioni: una è quella di richiedere l'immagine al WS, salvarla sul file system e referenziarla dalla pagina HTML. In questo caso, essendo immagini "dinamiche" che non ha senso renderle persistenti una volta mostrata la pagina, si dovrebbe prevedere anche un meccanismo di eliminazione. Molto più semplicemente si potrebbe far sì che la pagina HTML referenzi sempre una url specifica (per esempio immagini/random.jpg) ma a quella url non corrisponde alcuna immagine sul file system ma ad essa risponde una servlet! In questo caso la servlet dovrebbe preoccuparsi di mandare sulla response l'immagine letta dal WS. Resta un problema: come comunicare alla servlet l'ID a cui l'immagine si riferisce? Molto semplicemente si potrebbe usare un parametro sulla query string, per esempio immagini/random.jsp?id=valore. Ecco il codice da inserire nella pagina HTML che contiene la form mostrata in precedenza:

```
<img src="immagini/random.jpg?id=<%= id %>"
    alt="Testo generato casualmente" />
```

Ecco, invece, la servlet che reperisce l'immagine dal

WS e la mette sulla request:

```
public class ImageServlet extends HttpServlet {
 protected void doGet(HttpServletRequest request,
  HttpServletResponse response)
  throws ServletException, IOException {
  CaptchaWebServiceServiceLocator loc = new
    CaptchaWebServiceServiceLocator();
  CaptchaWebService_PortType ws=null;
  try {
    ws = loc.getCaptchaWebService();
  } catch (ServiceException e) {
      e.printStackTrace();
  }
  String id = request.getParameter("id");
  setResponseHeader(response);
  ServletOutputStream responseOutStream
     = response.getOutputStream();
  responseOutStream.write(ws.getImage(id));
  responseOutStream.flush();
  responseOutStream.close();
 }
}
```

Il metodo *setResponseHeader* non fa altro che impostare lo header http affinché non venga messa in cache la pagina e venga impostato il mime-type corretto:

```
 private void setResponseHeader(
 HttpServletResponse response){
  response.setHeader("Cache-Control", "no-store");
  response.setHeader("Pragma", "no-cache");
  response.setDateHeader("Expires", 0);
  response.setContentType("image/jpeg");
 }
```

In questo modo se l'utente esegue un back sul browser, l'immagine viene comunque caricata nuovamente (e diversa!).
Ecco invece il mapping da inserire sul file WEB-INF/web.xml della webapp affinché la servlet risponda con l'immagine all'url immagini/random.jpg:

```
<servlet>
  <servlet-name>ImmagineRandom</servlet-name>
  <servlet-class>
    it.ioprogrammo.jcaptcha.ImageServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>ImmagineRandom</servlet-name>
  <url-pattern>/immagini/random.jpg</url-pattern>
</servlet-mapping>
```

Ecco la pagina risultato.jsp che verifica la correttezza della risposta:

```
<%
 String id = request.getParameter("id_ws");
 Boolean match = null;
 try{
  CaptchaWebServiceServiceLocator loc =
    new CaptchaWebServiceServiceLocator();
  CaptchaWebService_PortType ws =
                               loc.getCaptchaWebService();
  match = ws.isCorrect(id,
                request.getParameter("testo_random"));
 }catch(Exception e){
  e.printStackTrace();
 }
 if (match!=null && match.booleanValue()) {
%>
 Risposta corretta!
<%
} else {
%>
 Risposta errata!
<%
}
%>
```

Negli esempi allegati all'articolo questa seconda webapp si chiama CaptchaWsClient.war (in Figura 3 l'esempio della sua esecuzione).

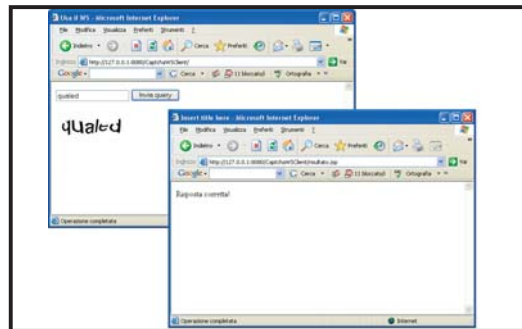



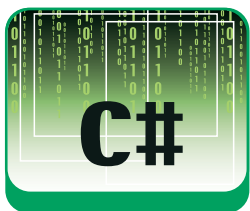
Fig. 4: *Il client di esempio*

CONCLUSIONI

L'articolo ha mostrato con quanta semplicità sia possibile fornire un servizio Web a siti che necessitano di tecniche CAPTCHA. In realtà si potrebbe estendere ulteriormente l'esempio realizzando politiche di sicurezza più evolute (per esempio usando uno dei metodi proposti dallo standard WS-Security). Inoltre potrebbe essere fornito un servizio personalizzato affinché le immagini generate possano provenire da un vocabolario, oppure con un set di caratteri personalizzato e così via.

Ivan Venuti

COMUNICAZIONI BASATE SUI CONTRATTI

SVILUPPARE SERVIZI SIGNIFICA PENSARE NON SOLO ALLA PROPRIA PIATTAFORMA DI SVILUPPO, MA GUARDARE A CONCETTI COME STANDARD E INTEROPERABILITÀ AL FINE DI CONSENTIRE L'UTILIZZO DELL'APPLICAZIONE CON QUALUNQUE TECNOLOGIA

In una qualsiasi architettura SOA (Service-Oriented) esiste un provider di servizi ed un consumer che, comunicando tra di loro, si scambiano continuamente informazioni strutturate secondo protocolli standard come, ad esempio, SOAP. L'utilizzo dello standard SOAP, però molto spesso non è sufficiente per garantire la correttezza della comunicazione. In questo caso, come possiamo garantire cbe il consumer ed il provider "parlino la stessa lingua"? Lo scambio di informazioni, per essere interoperabile, deve rispettare delle regole che non appartengono strettamente ad una tecnologia. Dobbiamo perciò definire i dati e gli schemi oggetto dei nostri servizi. Con questi presupposti nasce il concetto di "contratto". Nei Web Services tradizionali il contratto è generalmente costituito dal WSDL (Web Service Description Language), il cui compito è quello di descrivere ed esporre le caratteristiche di un servizio, tra cui le operazioni consentite e i dati o schemi trattati, proprio al fine di permettere ai client la corretta implementazione. In Windows Communication Foundation questo concetto viene rispettato pienamente e può essere implementato in diverse forme, ognuna con caratteristiche differenti e per questo adatta alle specifiche esigenze.
Nell'articolo affronteremo lo sviluppo di un client che utilizza un servizio per ricercare libri su un database remoto. Per ottenere una comunicazione efficiente, sia il consumer, sia il servizio, implementeranno lo stesso contratto, stabilito e concordato preventivamente.

I "CONTRATTI" DI WCF

Prima di affrontare e risolvere lo scenario descritto, cerchiamo di capire quali strumenti Windows Communication Foundation propone e quali possono risultarci utili per affrontare le diverse problematiche.
In Windows Communication Foundation definisce principalmente tre tipi di contratti:

- Service Contracts: contiene l'indicazione i metodi, chiamati operation, ed i parametri che un client deve rispettare per comunicare con il servizio;
- Data Contracts: è il tipo di contratto che definisce gli schemi che rappresentano i tipi di dati scambiati tra le entità;
- Message Contracts: consentono di lavorare direttamente sul messaggio SOAP, impostando le proprietà che appartengono all'header e quelle che appartengono al body;

Ogni tipo di contratto assolve ad una funzione differente. Se la nostra esigenza è semplicemente quella di definire le operazioni esposte, un concetto essenziale per un servizio, allora è sufficiente utilizzare il ServiceContract. Se, invece, abbiamo tutta l'intenzione di definire accuratamente le nostre entità, descrivendole sotto forma di schemi, allora dobbiamo utilizzare il DataContract . Le due tipologie erano già presenti nell'implementazione tradizionale, attraverso gli ASP.NET Web Service, utilizzando gli attributi WebService e WebMethods, per definire la classe di servizio e le sue operazioni, e la serializzazione, con l'utilizzo di XmlSerializer, per la definizione in schemi ed il controllo delle entità oggetto di scambio. Si aggiunge, ora, una nuova forma di contratto che permette di lavorare direttamente sul messaggio, modellandone le proprietà in modo da indirizzarle nell'header o nel body della busta SOAP a seconda delle esigenze: il Message Contract.
Tutti i contratti vengono definiti sfruttando le potenzialità degli attributi. E' sufficiente decorare interfacce, classi, proprietà e metodi con gli appositi attributi, per ottenere il contratto desiderato. E' importante chiarire che le informazioni indicate nei contratti trovano tutte una loro diretta corrispondenza nel WSDL, IL contratto che modella il nostro servizio.

IL SERVIZIO BOOKSEARCH

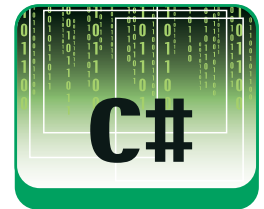
In Windows Communication Foundation l'implementazione di un servizio dipende direttamente dalla definizione del suo contratto. Il contratto, o WSDL, realizza la sua implementazione nello specifico linguaggio attraverso le interfacce. In C#, ad esempio, è possibile definire una interfaccia e decorarla con gli attributi ServiceContract e OperationContract per poi ottenere il contratto di un servizio. Proviamo a capire come funziona il meccanismo di definizione dei contratti creando una libreria di servizi. Apriamo, quindi, Visual Studio 2005 e creiamo un nuovo progetto Class Library. Qui aggiungiamo un file chiamato IBookSearch ed inseriamo il codice seguente:

```
[ServiceContract(Namespace="http://
        www.ioprogrammo.it/2006/06/BookSearch")]
public interface IBookSearch
{
        [OperationContract()]
        string FindByName(string bookName);
}
```

Il codice riportato specifica esattamente le operazioni che il nostro servizio deve implementare per poter poi essere consumato dai client che ne faranno sottoscrizione. L'interfaccia diventa, nella nostra applicazione, il contratto del servizio. Proviamo ad implementare l'interfaccia definita aggiungendo un nuovo file, chiamato BookSearch.cs, ed inserendo il codice seguente:

```
public class BookSearch : IBookSearch
{
        public string FindByName(string bookName)
{
// qui l'implementazione del servizio che ritorna
// la descrizione del libro richiesto
// il codice completo è disponibile sul CD allegato
}
}
```

Ho volutamente omesso, nell'articolo, l'implementazione interna del metodo FindByName per motivi di spazio. Il codice presentato rappresenta l'implementazione di un servizio che accetta in input il nome del libro e ne verifica la presenza all'interno dello store. Da notare è come non sia necessario definire null'altro che l'interfaccia da implementare per realizzare un servizio. Questo significa che siamo in grado di implementare l'interfaccia anche più volte ed in maniera del tutto differente. Possiamo ora mettere in hosting il nostro servizio, ad esempio, su una Console Application. Creiamo quindi un nuovo progetto console, aggiungiamo un riferimento alla libreria dei servizi precedentemente creata ed inseriamo nella procedura di avvio Main il codice per esporre il servizio:

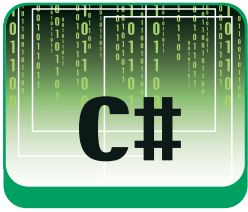
```
// creo l'hosting del servizio
ServiceHost host = new
ServiceHost(typeof(Library.BookSearch));
// creo l'hosting del servizio
host.AddServiceEndpoint(typeof(Library.IBookSearch),
        new basicHttpBinding(),
        "http://localhost:9001/BookSearch");
// apro il canale per mettere in ascolto il servizio
host.Open();
// attendo che arrivino le richieste
Console.ReadLine();
// chiudo il canale
host.Close();
```

SERVIZI SENZA CONTRATTI

Il modello basato sui contratti consente di trattare facilmente ed in modalità strongly-typed i messaggi scambiati tramite i servizi. Cosa succede, però, se non conosciamo a priori il contenuto del messaggio? In che modo ci dobbiamo comportare? Anche in questo caso esiste un particolare servizio che si occupa di leggere il contenuto del messaggio SOAP senza la necessità di definirlo a priori. Il servizio non utilizza nessun contratto, ma accetta in ingresso un solo parametro di tipo Message e restituisce in output un oggetto dello stesso tipo. Riprendendo l'esempio dell'articolo, potremmo avere il contratto del servizio così strutturato:

```
[ServiceContract()]
public interface IBookSearch
{
[OperationContract()]
Message FindByName(Message
                        request);
}
```

il codice riportato crea un hosting che si occupa di esporre l'implementazione del servizio, contenuta nella classe Library.BookSearch, attraverso un endpoint composto rispettivamente da un address http://localhost:9001/BookSearch, un binding identificato con l'utilizzo dell'oggetto basicHttp Binding e un contratto definito nella nostra interfaccia Library.IBookSearch. Il consumer del servizio apre invece un canale (channel) per attivare la comunicare:

```
// creo il channel
ChannelFactory<Library.IBookSearch> channel = new
ChannelFactory<Library.IBookSearch>("BookSearchE
                                dpointConfig ");
// creo il proxy
Library.IBookSearch proxy = channel.CreateChannel();
// utilizzo il metodo per la traduzione
string descrizione = proxy.FindByName
                        ("Programming Indigo");
```

Il codice dimostra come il consumer comunica con il provider di servizi attraverso l'apertura di un

canale utilizzando la classe ChannelFactory. Nel caso viene utilizzata una sezione del file di configurazione per definire le caratteristiche dell'endpoint da raggiungere. Il codice completo è disponibile sul cd allegato alla rivista.

Programming Indigo
David Pallmann
(Microsoft Press) ISBN
0-7356-2151-9 (2005)

UN SERVIZIO AVANZATO CON IL DATACONTRACT

Il servizio implementato nel precedente paragrafo accetta tipi semplici sia come parametri sia come valori di ritorno. Molto spesso, però, abbiamo la necessità di scambiare informazioni più complesse e diversamente strutturate, eventualmente controllandone la serializzazione. Se, ad esempio, volessimo ottenere maggiori informazioni sul libro cercato, come ad esempio il prezzo ed il codice ISBN, abbiamo l'esigenza di utilizzare un particolare tipo di contratto: il DataContract. Esso consente di serializzare l'entità Book e contemporaneamente di ottenere uno schema dettagliato per descriverne al meglio le proprietà o le variabili locali che intendiamo trasferire. Per questo scopo definiamo l'entità Book utilizzando gli attributi DataContract e DataMember:

```
[DataContract(Name="bookInfo")]
public class BookInfo
{
        [DataMember(Name="name")]
        public string Name;
        [DataMember(Name="description")]
        public string Description;
        [DataMember(Name="price")]
        public decimal Price;
        [DataMember(Name="isbn")]
        public string ISBN;
}
```

Modifichiamo il ServiceContract aggiungendo un nuovo OperationContract che consente di ottenere un servizio avanzato:

```
[ServiceContract(Namespace="http://
        www.ioprogrammo.it/2006/06/BookSearch")]
public interface IBookSearch
{
        [OperationContract()]
        BookInfo FindByName(string bookName);
}
```

di conseguenza modifichiamo l'implementazione del servizio. Provando ad eseguire il tutto otteniamo un risultato come quello visualizzato in FIGURA .

MESSAGECONTRACT

Alcuni scenari, però, potrebbe prevedere lo scambio di messaggi che non necessariamente includono dei particolari tipi o entità, ma richiedono un maggior controllo diretto sul messaggio. Con il MessageContract possiamo intervenire direttamente sulla definizione delle proprietà per indicare se esse devono far parte dell'intestazione (header) o del corpo (body) del messaggio. Questo controllo avviene sempre in maniera dichiarativa utilizzando gli attributi MessageContract per l'indicazione del contratto, MessageHeader e MessageBody per la definizione del comportamento delle proprietà o, eventualmente, delel variabili locali rispettivamente al messaggio SOAP. Ipotizziamo l'uso del message contract per implementare un servizio che richiede la fornitura di informazioni supplementari come come, ad esempio, un identificativo univoco del client.

```
[MessageContract()]
public class FindByNameRequestMessage
{
        [MessageHeader(Name="clientId")]
        public string ClientID;
        [MessageBody(Name="bookName")]
        public string BookName;
}

[MessageContract()]
public class FindByNameResponseMessage
```

SVILUPPARE CON WCF. COSA CI SERVE?

Per cominciare a sviluppare servizi con Windows Communication Foundation dobbiamo scaricare ed installare le seguenti componenti rispettando l'ordine indicato:

∑ Visual Studio 2005 – E' l'ambiente di sviluppo per la produzione di applicazioni basate sul .NET 2.0. Rappresenta comunque un componente facoltativo poiché le applicazioni sono compilabili anche da linea di comando. E' anche possibile utilizzare le versioni express di visual studio scaricabili da questo indirizzo: http://msdn.microsoft.com/vstudio/express/default.aspx

- **WinFX Runtime Components – Consentono l'esecuzione delle applicazioni WinFX. Sono scaricabili dall'indirizzo: http://www. microsoft.com/ downloads/details.aspx?FamilyId=F51C4D96-9AEA-474F-86D3-172BFA3B828B&displaylang=en**
- **Windows SDK – Include documentazione, esempi, tool e ambienti per lo sviluppo sia di applicazioni Windows native che basate sul WinFX. L'SDK è scaricabile dall'indirizzo: http://www.microsoft.com/downloads/details.aspx?FamilyId=9BE1FC7F-0542-47F1-88DD-61E3EF88C402&displaylang=en**
- **"Orcas" WinFX Development Tools – Fornisce funionalità aggiuntive all'IDE di sviluppo Visual Studio 2005, come l'intellisense per XAML e i project templates per Windows Workflow Foundation e Windows Communication Foundation. E' un componente opzionale, dipende se si utilizza Visual Studio come ambiente di sviluppo, e può essere scaricato da questo indirizzo: http://www.microsoft.com/downloads/details.aspx?FamilyId=AD0CE56E-D7B6-44BC-910D-E91F3E370477&displaylang=en**

```
{
        [MessageHeader(Name="clientId")]
        public string ClientID;
        [MessageBody(Name="bookName")]
        public BookInfo Book;
}
```

Il codice riportato imposta due tipi di MessageContract. Il primo definisce il messaggio soap che il consumer invia al servizio, mentre il secondo specifica la risposta del servizio verso il consumer. Ovviamente il messaggio non è limitato ai tipi semplici, ma può anche essere utilizzato in combinazione con gli altri contratti, come il DataContract. E' così che realizziamo una nuova versione del servizio che fa uso, appunto, del MessageContract:

```
[ServiceContract(Namespace="http://
        www.ioprogrammo.it/2006/06/BookSearch")]
public interface IBookSearch
{
        [OperationContract()]
        FindByNameResponseMessage
    FindByName(FindByNameRequestMessage request);
}
```

Provando ad intercettare il messaggio SOAP ci accorgiamo come questo viene trattato e serializzato. In FIGURA vediamo un esempio di messaggio.

CONTRACT VERSIONING

Nella stesura dell'articolo siamo intervenuti più volte nel nostro contratto originale per modificarne le caratteristiche in base alle mutate esigenze. Modificando i parametri del metodo FindByName abbiamo violato una delle regole della Service-Oriented Architecture: Services are Autonomous. In pratica i client che prima implementavano quel contratto, ora non saranno più in grado di comunicare con il servizio, proprio a causa delle sue mutate caratteristiche. Questo ci porta a dover gestire il versioning, che può aver luogo su due diversi livelli:

- A livello di ServiceContract, quando ad esempio vengono rimosse delle operation oppure ne vengono modificati o aggiunti parametri
- A livello di DataContract, quando vengono modificate, rimosse o aggiunte proprietà

Nel primo caso è importante tenere presente che non possiamo eseguire l'overload dei metodi, ma se proprio non ne possiamo fare a meno, dobbiamo specificare la proprietà name che trasformerà la action nel WSDL. In qualsiasi caso è fondamentale mantenere invariato il servizio precedentemente creato. Il nostro esempio sarebbe così:

```
[ServiceContract(Namespace="http://
        www.ioprogrammo.it/2006/06/BookSearch")]
public interface IBookSearch
{
        [OperationContract(Name = "FindByName")]
        string FindByName(string bookName);
        [OperationContract(Name =
                                "FindByNameV2")]
        BookInfo FindByName(string bookName);
}
```

Il versioning nel DataContract riguarda principalmente l'aggiunta di nuovi membri pubblici. Se infatti dobbiamo aggiungere un nuovo membro Price al nostro oggetto Book, dobbiamo garantire la compatibilità con i precedenti consumer. Ci viene in soccorso la proprietà IsRequired. Se settata a false, infatti, rende facoltativa la presenza della nuova proprietà, un po' come la keywork optional di visual basic. In questo scenario la classe Book sarebbe così:

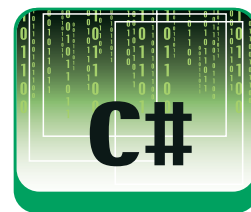
```
[DataContract(Name="bookInfo")]
public class BookInfo
{
        [DataMember(Name="name")]
        public string Name;
        [DataMember(Name="description")]
        public string Description;
        [DataMember(Name="price")]
        public decimal Price;
        [DataMember(Name="isbn")]
        public string ISBN;
        [DataMember(Name="language",
                            IsRequired="false"))]
        public string Language;
}
```

E' importante anche il ruolo del namespace, che diventa il mezzo per distinguere i contratti nel WSDL. Nel caso in cui, infatti, abbiamo la necessità di rendere obbligatoria la nuova proprietà, la soluzione è quella di creare una nuova interfaccia definita all'interno di un diverso namespace.

CONCLUSIONI

Nell'articolo abbiamo visto come l'infrastruttura di Windows Communication Foundation consente di rispondere agevolmente alle esigenze che riguardano le attuali architetture service-oriented. Garantendo al contempo interoperabilità fra le varie piattaforme. Il futuro senza dubbio non può prescindere da questo tipo di applicazione.

Fabio Cozzolino

SUL WEB

Windows Communication Foundation: Sito Ufficiale
http://www.windowscommunication.net

Windows Communication Foundation e Windows Vista
http://msdn.microsoft.com/windowsvista/connected/#wcf

Il mio blog
http://dotnetside.org/blogs/fabio

L'AUTORE

Fabio Cozzolino sviluppa software da circa 10 anni. Si è appassionato al .NET Framework fin dalla prima versione sviluppando principalmente applicazioni web, raggiungendo la certificazione MCAD. Negli ultimi due anni si è dedicato alla produzione di software in architetture distribuite e service-oriented. E' cofondatore dello user group DotNetSide con il quale organizza eventi periodici dedicati al mondo .NET.

IO PROGRAMMO BY EXAMPLE

IMPARA A PROGRAMMARE IN MODO PRATICO E DIVERTENTE, CON GLI ESEMPI PASSO PASSO CHE TI GUIDANO ALLA COSTRUZIONE DEL CODICE

VUOI INVIARE UN ESEMPIO?

Se sei un programmatore esperto ed hai risolto un problema, puoi aiutare gli altri pubblicando il tuo codice. Proponi i tuoi esempi scrivendo a ioprogrammo@edmaster.it

Come contattarci?

Alla nostra redazione arriva spesso un considerevole numero di email riguardante temi specifici. Per consentirci di rispondere velocemente e in modo adeguato alle vostre domande abbiamo elaborato una FAQ – Frequently Ask Question – o risposte alle domande frequenti in italiano, di modo che possiate indirizzare le vostre richieste in modo mirato.

Problemi sugli abbonamenti

Se la tua domanda ha a che fare con una delle seguenti:

- Vorrei abbonarmi alla rivista, che devo fare?
- Sono un abbonato e non ho ricevuto la rivista, a chi devo rivolgermi?
- Sono abbonato ma la posta non mi consegna regolarmente la rivista, a chi devo rivolgermi?

Contatta **abbonamenti@edmaster.it** specificando che sei interessato a ioProgrammo. Lascia il tuo indirizzo email e indica il numero dal quale vorresti far partire l'abbonamento. Verrai contattato al più presto. Oppure puoi chiamare lo **02 831212**

Problemi sugli allegati

Se riscontri un problema del tipo:

- Ho acquistato ioProgrammo ed il Cdrom al suo interno non funziona. Chi me lo sostituisce?
- Ho acquistato ioProgrammo ma non ho trovato il cd/dvd all'interno, come posso ottenerlo?
- Vorrei avere alcuni arretrati di ioProgrammo come faccio?

Contatta **servizioclienti@edmaster.it**
Non dimenticare di specificare il numero di copertina di ioProgrammo e la versione: con libro o senza libro. Oppure telefona allo **02 831212**

Assistenza tecnica

Se il tuo problema è un problema di programmazione del tipo:

- Come faccio a mandare una mail da PHP?
- Come si instanzia una variabile in c++?
- Come faccio a creare una pagina ASP.NET

o un qualunque altro tipo di problema relativo a tecniche di programmazione, esplicita la tua domanda sul nostro forum: **http://forum.ioprogrammo.it**, uno dei nostri esperti ti risponderà. Le domande più interessanti saranno anche pubblicate in questa rubrica.

Problemi sul codice all'interno del CD

Se la tua domanda è la seguente

- Non ho trovato il codice relativo all'articolo all'interno del cd

Consulta la nostra sezione download all'indirizzo **http://cdrom.ioprogrammo.it**, nei rari casi in cui il codice collegato ad un articolo non sia presente nel cdrom, senza dubbio verrà reso disponibile sul nostro sito.

Idee e suggerimenti

Se sei un programmatore esperto e vuoi proporti come articolista per ioProgrammo, oppure se hai suggerimenti su come migliorare la rivista, se vuoi inviarci un trucco suggerendolo per la rubrica tips & tricks invia una email a **ioprogrammo@edmaster.it**

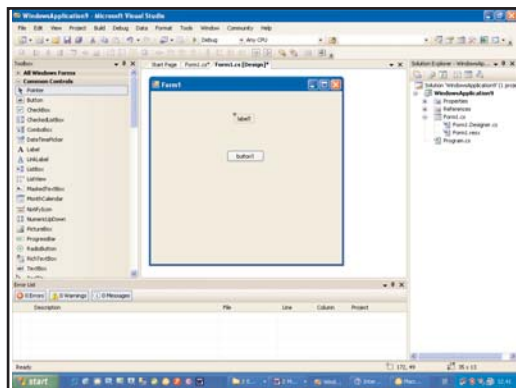
COME POSSO SAPERE SE UNA CONNESSIONE È ATTIVA?

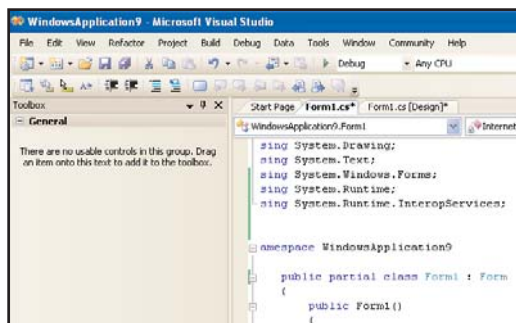
USIAMO LA LIBRERIA WININET.DLL PER CAPIRE SE LA LAN O IL MODEM SONO CONNESSI IN UN DETERMINATO MOMENTO

C#

VISUAL BASIC.NET

FACCIAMOLO IN C#

1 Creiamo un nuovo progetto di tipo Windows Application e sulla form trasciniamo un bottone e una label dalla palette degli strumenti

2 Spostiamoci nella sezione relativa al codice e aggiungiamo le seguenti righe poco sotto quelle della dichiarazione della classe

```
[DllImport("wininet.dll")]
    private static extern bool
  InternetGetConnectedState(out int Description, int
                                      ReservedValue);
```

3 Aggiungiamo un nuovo metodo che sfrutti il riferimento esterno appena dichiarato

```
  public static bool IsConnectedToInternet()
                                          {
      int Description
return InternetGetConnectedState(out
Description, 0);
    }
```

4 Portiamoci di nuovo sulla form e clicchiamo due volte sul bottone per generare il metodo che gestirà l'evento OnClick

```
private void button1_Click(object sender,
                                  EventArgs e)
     {
        label1.Text =
              IsConnectedToInternet().ToString();}
```

COME FUNZIONA?

La DLL wininet esporta il metodo InternetGet ConnectedState. Questo metodo prende come input due parametri. Un parametro che contiene la descrizione del tipo di connessione da testare: INTERNET_CONNECTION_LAN,INTERNET_CONNECTION_MODEM,INTERNET_CONNECTION _PROXY,INTERNET_ CONNECTION_OFFLINE e una costante riservata che possiamo considerare sempre uguale a 0. Ricaviamo il nome della connessione da testare attraverso la parola chiave Out alla quale passiamo inizialmente una descrizione vuota. InternetGetConnect State restituisce un valore boolean se la connessione è attiva. Nella gestione del click non facciamo altro che farci stampare questo valore a video.

CHE VUOL DIRE CONNESSIONE ATTIVA?

Per connessione Attiva non si intende che il cavo sia connesso e che la connessione sia ben configurata, ma si intende proprio capire se a livello software il mezzo di trasporto è predisposto al funzionamento. Ovvero questa procedura restituisce false se la connessione alla lan viene disattivata dal pannello di controllo, ma continua a restituire true se la connessione è attiva e il cavo scollegato

GLI IMPORT DA USARE CON C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime;
using System.Runtime.InteropServices;
```

FACCIAMOLO CON VISUAL BASIC

1 Creiamo un nuovo progetto di tipo Windows Application e sulla form trasciniamo un bottone e una label dalla palette degli strumenti

```
Private Declare Function
        InternetGetConnectedState Lib "wininet" _
                (ByVal Description As Integer, ByVal
        ReservedValue As Integer) As Boolean
```

2 Riscriviamo la Function che richiama la DLL in questione

```
Public Function IsConnectedToInternet() As Boolean
        Dim Description As Integer
        Return
            InternetGetConnectedState(Description, 0)
    End Function
```

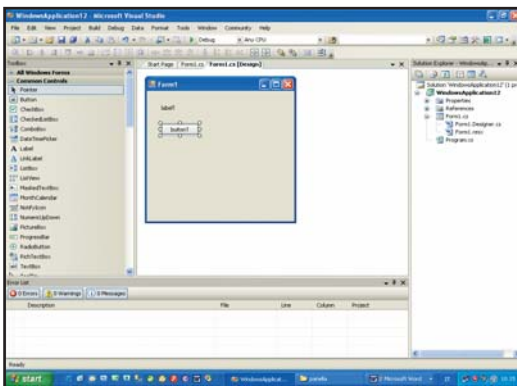
3 Infine riscriviamo il metodo che gestisce il click sul bottone. La sintassi è abbastanza semplice, si tratta di modificare il testo della label

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles Button1.Click
        Label1.Text =
                    IsConnectedToInternet.ToString()
    End Sub
```

COME POSSO ACCEDERE AL REGISTRO DI SISTEMA?

E' POSSIBILE FARLO ATTRAVERSO LA CLASSE REGISTRY DI MICROSOFT.WIN32. VEDIAMO COME RECUPERARE INFORMAZIONI SULLA CPU ATTRAVERSO IL REGISTRY

FACCIAMOLO IN C#

1 Creiamo una nuova applicazione di tipo Windows Application e trasciniamo sulla form un bottone e una label

2 Clicchiamo due volte sul bottone per generare il codice di gestione del'evento OnClick e scriviamolo come segue:

```
private void button1_Click(object sender,
                                    EventArgs e)
    {
        label1.Text = get_cpu();
    }
```

3 Scriviamo la il metodo get_cpu()

```
private string get_cpu() {
        RegistryKey _Key;
        RegistryKey _LocalMachine =
                            Registry.LocalMachine;
        _Key=_LocalMachine.OpenSubKey(");
        Object obj=_Key.GetValue("Identifier");
            return obj.ToString();
    }
```

COME FUNZIONA?

Sfruttiamo la classe RegistryKey per instanziare due oggetti di questo tipo. Il secondo oggetto lo inizializziamo facencdolo puntare al registro locasle, preleviamo poi la chiave che ci interessa attraverso il metodo OpenSubKey, infine recuperiamo il valore della key attraverso il metodo GetValue. Infine valorizziamo la funzione.

GLI IMPORT DA USARE CON C#

All'inizio del codice è necessario aggiungere

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.Win32;
```

FACCIAMOLO CON VISUAL BASIC

1 Ricreiamo la form così come abbiamo fatto per C# e clicchiamo due volte sul bottone per gene-

rare il codice di gestione dell'evento OnClik. Il codice sarà il seguente

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles Button1.Click
        Label1.Text = get_cpu()
    End Sub
un bottone
```

2 Scriviamo il metodo get_cpu come segue

```
Private Function get_cpu() As String
        Dim _Key As RegistryKey
        Dim _LocalMachine As RegistryKey
        Dim obj As Object
        _LocalMachine = Registry.LocalMachine
        _Key =
    _LocalMachine.OpenSubKey("HARDWARE\\DESCRIP
                   TION\\System\\CentralProcessor\\0")
        obj = _Key.GetValue("Identifier")
        Return obj.ToString
    End Function
== Box gli import da usare in Visual Basic ==
All'inzio del codice è necessario aggiungere
Imports Microsoft.Win32
```

GLI IMPORT DA USARE IN VISUAL BASIC

All'inzio del codice è necessario aggiungere

Imports Microsoft.Win32

COME POSSO FARE L'UPLOAD DI UN FILE TRAMITE IL WEB

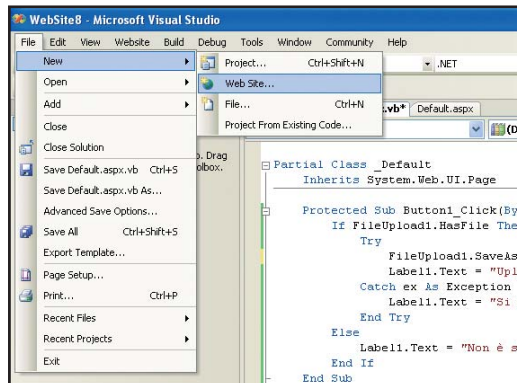
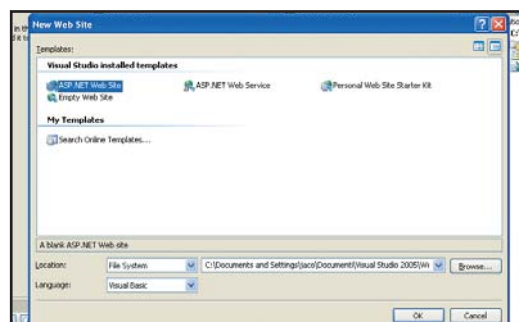
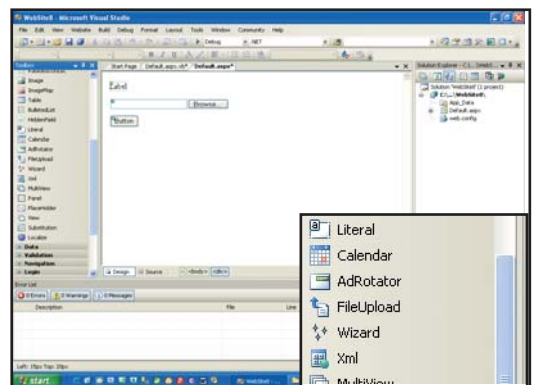
PHP E ASP.NET HANNO APPROCCI PROFONDAMENTE DIVERSI. VEDIAMO COME RISOLVERE IL PROBLEMA CON I DUE LINGUAGGI

ASP.NET

C#

PHP

FACCIAMOLO IN ASP.NET

1 Creiamo un nuovo progetto Web

2 Nella maschera che segue poniamo attenzione nel selezionare "Asp.NET Web Site" e in basso il nome del linguaggio "Visual Basic"

3 Realizziamo una pagina web trascinando su di essa un componente di tipo FileUpload, un bottone e una label

4 Clicchiamo due volte sul bottone per creare lo scheletro del codice di gestione dell'evento OnClick e modifichiamolo come segue

```
Protected Sub Button1_Click(ByVal sender As
        Object, ByVal e As System.EventArgs) Handles
                                        Button1.Click
        If FileUpload1.HasFile Then
            Try
            FileUpload1.SaveAs("C:\uploads" &
                                  FileUpload1.FileName)
        Label1.Text = "Upload avvenuto con successo!!"
            Catch ex As Exception
              Label1.Text = "Si è verificato un errore!"
            End Try
```

```
        Else
   Label1.Text = "Non è stato selzionato nessun file."
        End If
      End Sub
```

COME FUNZIONA?

Il componente FileUpload è un'innovazione del Framework 2.0, il suo utilizzo facilita di molto la gestione dell'upload di file sul web. Il funzionamento sufficientemente semplice. Nel primo if utilizziamo il metodo HasFile per sapere se l'utente ha effettivamente selezionato un file da uploadare, se non lo ha fatto modifichiamo in modo opportuno il testo della label. Se l'utente ha selezionato qualche file proviamo ad effettuare l'upload attraverso il metodo SaveAs, se tutto funziona regolarmente modifichiamo opportunamente il testo della label, viceversa intercettiamo l'eccezione e scriviamo un messaggio d'errore

FACCIAMOLO IN C#

1 Ripercorriamo i passi da uno a quattro avendo cura di modificare il nome del linguaggio nella seconda dialog box. Clicchiamo due volte sul bottone e modifichiamo il codice come segue

```
protected void Button1_Click(object sender
                                        ,EventArgs e)
    {
        if (FileUpload1.HasFile) {
            try {
                FileUpload1.SaveAs("C:\\uploads\\" +
                                        FileUpload1.FileName);
        Label1.Text = "Upload avvenuto con successo!!";
            } catch (Exception ex) {
        Label1.Text = "Si è verificato un errore!";
            }
        } else {
        Label1.Text = "Non è stato selzionato nessun file.";
        }
    }
```

FACCIAMOLO IN PHP

1 Creiamo un file chiamato index.php e al suo interno inseriamo il seguente script

```
<html>
<head>
<script type="text/javascript">
function hasfile() {
            if
        (document.getElementById('userfile').value !="") {
                    contenitore.submit();
            } else {
document.getElementById('feedback').innerHTML=
                            devi selezionare almeno un file";
            }
}
</script>
</head>
<body>
<p id=feedback></p>
<form id=contenitore enctype="multipart/
                    form-data" action="" method=post>
<input name="userfile" id="userfile" type="file">
<input value="Invia" type=button onclick="hasfile()">
</form>
<?
 $uploaddir = '/home/fabio/public_html/upl2/';
 $uploadfile = $uploaddir .
                    basename($_FILES['userfile']['name']);
 echo "<pre>";
 if
     (move_uploaded_file($_FILES['userfile']['tmp_name'
                                        ], $uploadfile)) {
    echo '<script type="text/javascript">';
    echo '
     document.getElementById(\'feedback\').innerHTML=
                            "File Uploadato con successo"';
    echo '</script>';
 } else {
     echo '
     document.getElementById(\'feedback\').innerHTML=
  "Si è verificato un errore contattarel\'amministratore"';
 }
?>
```

COME FUNZIONA IN PHP?

Abbiamo dovuto ricreare comportamenti che in .NET sono implementati direttamente nel framework, con un miscuglio di PHP e JavaScript. In realtà la parte JavaScript serve soltanto a validare la form. Prima di effettuare il submit si controlla che ci sia almeno qualcosa di scritto nel campo file, dopodiché si effettua l'upload con i metodi previsti da PHP. Se tutto va a buon fine viene cambiato il campo di feedback ancora tramite javascript

COME POSSO CHIUDERE TUTTI I FOGLI DI EXCEL?

Un metodo abbastanza semplice è il seguente:

```
Public Sub CloseAll()
Dim Wb As Workbook
SaveAll
For Each Wb In Workbooks
If Wb.Name <> ThisWorkbook.Name Then
Wb.Close savechanges:=True
End If
Next Wb
ThisWorkbook.Close savechanges:=True

End Sub
```

COME POSSO AGGIUNGERE UN SUONO A UN BOTTONE IN EXCEL?

OCCORRE RICHIAMARE UNA DLL CONTENUTA NEL SISTEMA. LA PROCEDURA TUTTAVIA È ABBASTANZA SEMPLICE

1 Abilitiamo la palette dei moduli cliccando su visualizza/barre degli strumenti/moduli

2 Dalla barra dei moduli selezioniamo un bottone

3 Disegniamo il bottone sul foglio di lavoro utilizzando il mouse. Appena finito comparirà anche una dialog box

4 Dalla Dialog Box delle macro selezioniamo nuovo

5 Modifichiamo il codice di gestione

```
Declare Function sndPlaySound32 Lib "winmm.dll"
                                        Alias _
    "sndPlaySoundA" (ByVal lpszSoundName AsString, _
    ByVal uFlags As Long) As Long
Sub Pulsante1_Clic()
CallsndPlaySound32("c:\programmi\media\
chimes.wav", 0)
End Sub
```

COME POSSO RECUPERARE IL NOME DEI FOGLI ATTIVI?

Per recuperare il nome del primo foglio presente nella cartella:

```
Public Function FirstSheetName()
   FirstSheetName = Sheets(1).Name
End Function
```

Per ottenere un array contenente tutti i nomi dei fogli presenti

```
Public Function AllSheetNames()
   Dim Arr() As String
   Dim I as Integer
   Redim Arr(Sheets.Count-1)
   For I = 0 To Sheets.Count - 1
      Arr(i) = Sheets(I+1).Name
   Next I
   AllSheetNames = Arr
   ' return a row array OR

AllSheetNames =
Application.Worksheetfunction.Transpose(Arr)
  ' return a column array
End Function
```

COME POSSO EVIDENZIARE LA CELLA ATTIVA DI EXCEL?

UTILIZZIAMO UNA MACRO E IL VISUAL BASIC SCRIPT EDITOR PER GESTIRE L'EVENTO SHEETSELECTIONCHANGE

1 Apriamo un nuovo foglio di lavoro e dal menu strumenti selezioniamo Macro/Visual Basic Editor

2 Dallo script editor clicchiamo due volte su "This Workbook", comparirà uno spazio per l'immissione del codice

3 Selezioniamo dal menu a tendina di destra "WorkBook" e dal menu a tendina di sinistra "Sheet Selection Change" comparirà lo scheletro di gestione del codice. Il codice da aggiungere è il seguente

```
Private Sub Workbook_SheetSelectionChange(ByVal
            Sh As Object, ByVal Target As Range)
    Static OldRange As Range
    On Error Resume Next
        Target.Interior.ColorIndex = 6 ' yellow -
                                  change as needed
        OldRange.Interior.ColorIndex =
                                  xlColorIndexNone
    Set OldRange = Target
End Sub
```

4 Torniamo al foglio di lavoro cliccando sul simbolo in alto a sinistra

COME POSSO CALCOLARE L'ETÀ DI UNA PERSONA?

Il metodo più semplice è usare la funzione datediff come segue

= ") & " anni, " & DATEDIF(A1,NOW(),"ym") & " mesi, " & DATEDIF(A1,NOW(),"md") & " giorni"

che ritorna una stringa del tipo 33 anni, nove mesi, 18 giorni Ma possiamo usare la stessa funzione anche in codice VBA come segue:

```
Function Age(Date1 As Date, Date2 As Date) As
                                        String
    Dim Y As Integer
    Dim M As Integer
    Dim D As Integer
    Dim Temp1 As Date
        Temp1 = DateSerial(Year(Date2),
                Month(Date1), Day(Date1)
    Y = Year(Date2) - Year(Date1) + (Temp1 > Date2)
    M = Month(Date2) - Month(Date1) - (12 * (Temp1 >
Date2))
    D = Day(Date2) - Day(Date1)
    If D < 0 Then
        M = M - 1
        D = Day(DateSerial(Year(date2),
Month(date2), 0)) + D
    End If
    Age = Y & " years " & M & " months " & D &
" days"
End Function
```

CHE COSA È RUBY?

PROBABILMENTE SI TRATTA DEL LINGUAGGIO IN PIÙ FORTE ASCESA DEL MOMENTO. VI MOSTRIAMO UN ESEMPIO SU COME CREARE UN GRAFICO IN EXCEL TRAMITE UNO SCRIPT SVILUPPATO IN RUBY

1 Dal menu start avviamo l'editor FreeRide

2 Non ci sono regole complesse da seguire per la scrittura di uno script in ruby. E' sufficiente scrivere il codice, per farlo avremmo potuto anche usare il notepad. Il nostro codice d'esempio è il seguente:

```
require 'win32ole'

ChartTypeVal = -4100;
excel = WIN32OLE.new("excel.application")
excel['Visible'] = TRUE;
workbook = excel.Workbooks.Add();
excel.Range("a1")['Value'] = 3;
excel.Range("a2")['Value'] = 2;
excel.Range("a3")['Value'] = 1;
excel.Range("a1:a3").Select();
excelchart = workbook.Charts.Add();
excelchart['Type'] = ChartTypeVal;

30.step(180, 10) do |rot|
    excelchart['Rotation'] = rot
end

excelchart2 = workbook.Charts.Add();
excelchart3 = workbook.Charts.Add();

charts = workbook.Charts
charts.each { |i| puts i }

excel.ActiveWorkbook.Close(0);
excel.Quit();
```

3 Salviamo lo script in una qualunque posizione sul disco ed eseguiamolo cliccando due volte sul file. Il risulatato sarà sorprendente. In figura vedete un'immagine statica, nella realtà il vostro grafico effettuerà una rotazione 3D

COME FUNZIONA?

Uno degli elementi di forza di ruby sta nella curva di apprendimento decisamente bassa. In questo script viene instanziato un oggetto Excel tramite una API Win32. Questo oggetto rappresenta in tutto e per tutto un foglio excel. A questo foglio viene per prima cosa aggiunta una cartella di lavoro, a seguire si riempiono tre celle con un valore e infine si aggiunge un grafico, valorizzato sulla base di una selezione effettuata sulle celle appena riempite. L'effetto rotazione si ottiene con un semplice ciclo di for.

DOVE POSSO TROVARE RUBY?

Nel CD allegato alla nostra rivista, oppure OnLine all'indirizzo http://www.rubycentral.com/downloads/index.html

CHE COSA È RUBYONRAILS

Sostanzialmente un framework i sviluppo per Ruby. Quello che è più interessante è che ha appena vinto il premio come miglior FrameWork dell'anno!

COME POSSO OTTENERE UN NOME DA UN IP?

In php è abbastanza semplice, è sufficiente utilizzare la funzione gethostbyaddr

```
<?php
$hostname=gethostbyaddr("151.99.125.2");
print $hostname; ?>
```

COME POSSIAMO MIGRARE UN DATABASE DA ACCESS A MYSQL?

UTILIZZIAMO IL MYSQL MIGRATION TOOLKIT, UNO STRUMENTO GRATUITO CHE CI CONSENTE DI TRASPORTARE DATI DA QUALUNQUE DB

1 Prima di tutto lanciamo il MySQL Migration Toolkit

2 Selezioniamo Direct Migration se il database Access è presente sulla nostra stessa macchina

3 Dal menu a tendina che compare selezioniamo MS Access

4 Riempiamo la form sottostante indicando il percorso del file da cui effettuare la migrazione

5 eseguiamo la stessa operazione indicando questa volta le credenziali per il database MySQL target

6 Clicchiamo su Next, verrà effettuato un test per verificare che tutte le condizioni per la migrazione siano soddisfatte

7 Clicchiamo ancora su Next e selezioniamo il nome per il database che dovrà contenere il DB Access importato

8 Se tutto è andato a buon fine tutti i check daranno esito positivo

9 Nella maschera seguente possiamo decidere quale tabelle migrare

10 Infine cliccando ancora Next fino alla fine il DB verrà completamente migrato

COME POSSO ESEGUIRE BACKUP PERIODICI DI MYSQL ?

ANCORA UNA VOLTA CI VENGONO IN AIUTO GLI STRUMENTI VISUALI MESSI A DISPOSIZIONE DA MYSQL AB

1 Iniziamo lanciando MySQL Administrator e inserendo le credenziali di autenticazione

2 Selezioniamo Backup dal menu a sinistra e New Project dal bottone in basso

3 Diamo un nome al progetto e stabiliamo di quali DB vogliamo mantenere un Backip

4 Dalla Tabsheet Advanced selezioniamo eventuali opzioni particolari per la tipologia di Backup. Lasciare le impostazioni di default è una buona soluzione

5 Dal menu Tools selezioniamo Options

6 Dalla Tabsheet "General Options" scegliamo "Store Password" e indichiamo il metodo "Obscured" come sistema per salvare le password per l'accesso al database di cui mantenere il Backup. Questa operazione ci consente anche di aumentare la sicurezza con cui manteniamo i dati all'interno del nostro computer

7 Infine indichiamo quali sono i giorni e le ore in cui desideriamo che il Backup venga effettuato

8 Indichiamo l'utente di sistema e diamo il via al backup. Verrà generato un file sql composto dal nome del file più la data in cui il backup è stato effettuato. Infine possiamo anche eseguire il primo Backup cliccando su Execute

STORED PROCEDURE IN CODICE MANAGED

UNA DELLE NOVITÀ PORTATE DALL'ACCOPPIATA .NET 2.0 E SQL SERVER 2005 È COSTITUITA DALLA POSSIBILITÀ DI SCRIVERE METODI DIRETTAMENTE DAL PROPRIO LINGUAGGIO PREFERITO PER POI UTILIZZARLI DALL'INTERNO DEL DB. VEDIAMO COME FUNZIONA...

Quante volte ci è capitato di dover scrivere delle query con calcoli matematici molto complessi e di non riuscire ad esprimerle come volevamo?
In questi casi, in genere, dovevamo estrarre i dati da SQL Server, "lavorarli" con il linguaggio di programmazione che stavamo utilizzando per la nostra applicazione e magari rinviarli a SQL Server. Questo determinava un aumento della complessità dell'applicazione ed uno spreco di risorse di rete per la comunicazione tra database ed applicazione.
T-SQL è un ottimo linguaggio per le comuni operazioni sui database ma non ha certamente la potenza espressiva di un linguaggio di programmazione come per esempio C#.
Una delle principali novità di SQL Server 2005 è l'integrazione con il CLR del framework 2.0: è, infatti, possibile scrivere stored procedure, trigger e funzioni in managed code.
Attualmente i linguaggi del framework supportati sono VB.Net, C# e managed C++.

I VANTAGGI E LA TECNOLOGIA

Cosa significa esattamente scrivere stored procedure in CLR? L'opportunità fornita da Visual Studio 2005 è la seguente:

- Creo una DLL utlizzando il linguaggio che mi è più familiare
- La DLL conterrà dei metodi che restituiscono dei valori sulla base di certe elaborazioni
- Posso elaborare i dati utilizzando i tipi e le strutture che più preferisco adottati dal mio linguaggio di riferimento
- L'elaborazione dei dati può essere semplice, ad esempio una differenza fra date temporali, o anche complessa, coinvolgendo a sua volta dati provenienti da un DB.
- Una volta che la mia DLL è pronta la registro in SQL Server, e posso richiamarla da qualunque progetto sotto forma di Stored Procedure. Una bella comodità non trovate?

UN ESEMPIO CONCRETO

Supponiamo di voler ricavare il numero della settimana corrente. Se utilizzassi direttamente T-SQL potrei adottare una formula del genere:

```
SELECT DATEPART(ww,'2006-24-05')
```

il valore risultante sarebbe 22, tuttavia il valore reale è 21. T-SQL non è preciso in questo! Se ripetiamo la stessa operazione in codice managed otteniamo

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles Button1.Click
    Dim dDate As Date
    dDate = #5/24/2006 12:14:00 PM#
    Label1.Text = DatePart("ww", dDate, Firs
            DayOfWeek.Monday, FirstWeekOfYear.
                        FirstFourDays).ToString
  End Sub
```

Il cui risultato è 21, come ci aspettavamo. Poter sviluppare le Stored Procedure in codice managed significa poter ricompilare questo metodo come DLL, registrarla in SQL Server e utilizzare la Stored Procedure corretta ogni qual volta essa ci possa servire.

SVILUPPARE IN CLR

Ciascun oggetto managed da noi creato ha un attributo che indica a SQL Server 2005 che tipo di componente CLR stiamo realizzando. Questi attributi possono essere ad esempio SqlStoredrocedure o SqlFunction.

Una semplice stored procedure potrebbe essere

```
public partial class StoredProcedures
{
    [Microsoft.SqlServer.Server.SqlProcedure]
    public static void listaUtenti()
    {
        SqlConnection myconn = new SqlConnection();
        myconn.ConnectionString =
                                    "Context Connection=true";
        myconn.Open();
        using (myconn)
        {
            SqlCommand mycomm =
                                    new SqlCommand();
            string sql = "select * from utenti";
            mycomm.Connection = myconn;
            mycomm.CommandText = sql;
            SqlDataReader myreader = null;
            using (myreader =
                                    mycomm.ExecuteReader())
            {
                SqlPipe p;
                p = SqlContext.Pipe;
                p.Send( myreader );
            }
        }
    }
};
```

Compilando il codice otteniamo un assembly che va installato su SQL Server.
La sintassi per l'installazione è la seguente

```
CREATE ASSEMBLY <nome_assembly>
FROM <percorso_dll>
```

Nel nostro caso scriveremmo

```
CREATE ASSEMBLY StoredProcedures
FROM c:\progetti\sqlserver\ CLRIntegration.dll
```

Quando integriamo un assembly in SQL Server, abbiamo la possibilità di specificare il livello di permessi per l'esecuzione tramite la sintassi WITH PERMISSION_SET = livello.
Il livello di sicurezza può assumere tre valori: SAFE, EXTERNAL e UNSAFE.
Una volta integrato l'assembly, possiamo utilizzare la dll per mappare le funzioni create in oggetti di SQL Server.
Per creare una stored procedure utilizziamo la sintassi

```
CREATE PROCEDURE <nome_procedura>
AS EXTERNAL NAME <identificatore_assembly>.
                                    <nome_tipo>.<metodo>
```

nel nostro esempio

```
CREATE PROCEDURE listaUtenti
AS EXTERNAL NAME CLRIntegration.
                                    StoredProcedures.listaUtenti
```

Allo stesso modo possiamo creare funzioni

```
CREATE FUNCTION <nome_funzione>
                                    (<lista_parametri>)
RETURNS <tipo_di_ritorno>
AS EXTERNAL NAME <identificatore_assembly>.
                                    <nome_tipo>.<metodo>
```

e trigger

```
CREATE TRIGGER <nome_trigger>
ON <tabella_o_view> <FOR|INSTEAD OF|AFTER>
                                    <INSERT|UPDATE|DELETE>
AS EXTERNAL NAME <identificatore_assembly>.
                                    <nome_tipo>.<metodo>
```

VISUAL STUDIO 2005

Il modo più naturale di scrivere oggetti managed per SQL Server 2005 è utilizzare Visual Studio 2005 che offre all'utente diverse funzionalità per semplificarne lo sviluppo e, soprattutto, il deploy.
Visual Studio 2005 ha, infatti, un nuovo tipo di progetto "SQL Server Project" che può essere utilizzato per scrivere oggetti managed per SQL Server 2005 dei quali riesce a creare automaticamente i template. Tramite Visual Studio 2005 è, inoltre, possibile fare il debug degli oggetti per testarne il corretto funzionamento.
Vediamo come utilizzarlo per creare stored procedure, funzioni e trigger in CLR.
Per gli esempi utilizzeremo la tabella Utenti ottenuta dalla seguente query DDL

```
CREATE TABLE [dbo].[Utenti](
            [idUtente] [int] IDENTITY(1,1) NOT NULL,
            [nome] [varchar](50) COLLATE Latin1_
                                    General_CI_AS NOT NULL,
            [cognome] [varchar](50) COLLATE
                                    Latin1_General_CI_AS NOT NULL,
            [email] [varchar](50) COLLATE
                                    Latin1_General_CI_AS NOT NULL,
            [scadenza] [varchar](50) COLLATE
        Latin1_General_CI_AS NOT NULL CONSTRAINT
                        [DF_Utenti_scadenza]  DEFAULT
                        (dateadd(year,(1),getdate())),
 CONSTRAINT [PK_Utenti] PRIMARY KEY CLUSTERED
                                    ( [idUtente] ASC )
)
```

CLR

Acronimo di Common Language Runtime, è il motore runtime del framework .Net
E' composto da due parti principali: il motore di esecuzione vero e proprio che permette di caricare in memoria gli assembly ed eseguirli, e da una libreria di classi che contengono una serie di classi per le funzionalità più disparate.

SQLCONTEXT

SqlContext rappresenta il contesto di esecuzione del managed code. Fornisce l'accesso agli oggetti SqlPipe SqlTriggerContext e WindowsIdentity. Ha inoltre una proprietà IsAvailable che ci dice se la connessione è disponibile.

Creiamo un nuovo progetto di tipo SQL Server Occorre creare una reference al database: selezioniamo il mone del server ed il database

Fig. 1: ***Creazione di un progetto "Sql Server"***

Fig. 2: ***Aggiunta di un riferimento a Sql Server***

STORED PROCEDURE

Supponiamo di voler creare una semplice procedura che seleziona gli utenti in base all'iniziale del cognome.
Per creare delle stored procedure, clicchiamo con il tasto destro del mouse sul nome del progetto, selezioniamo "Add" e poi "Stored procedure..."

Fig. 3: ***Creazione di una Stored Procedure***

Chiamiamo la procedura "FiltraUtenti" e scriviamo il codice per estrarre i dati.
Nella creazione di una Stored Procedure occorre ricordare tre cose fondamentali:
- la classe che la contiene deve essere public
- il metodo esposto deve essere public e static
- Il metodo esposto va "decorato" con l'attributo SqlProcedure
Vediamo il codice in C#

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
public partial class StoredProcedures
{
  [Microsoft.SqlServer.Server.SqlProcedure]
  public static void FiltraUtenti(SqlString iniziale)
  {
    using (SqlConnection connection = new SqlC
                 nection("context connection=true"))
    {
      connection.Open();
      string sql =
        "select * from utenti " +
        " where " +
        "  substring(cognome,1,1) = '" + iniziale.
                              ToString() + "'";
      SqlCommand command = new
                    SqlCommand(sql, connection);
      SqlDataReader r = command.
                              ExecuteReader();
      SqlContext.Pipe.Send(r);
    }
  }
};
```

e il corrispondente in Visual Basic.NET

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Data.SqlTypes
Imports Microsoft.SqlServer.Server
Partial Public Class StoredProcedures
  <Microsoft.SqlServer.Server.SqlProcedure()> _
  Public Shared Sub  FiltraUtenti (iniziale as
                                    SqlString)
    Using connection As New SqlConnection("context
                              connection=true")
      connection.Open()
      Dim sql As String = "select * from utenti " & _
              " where " & _
              "  substring(cognome,1,1) =
                                       '" & _
              iniziale.ToString & "'"
      Dim command As New SqlCommand
```

```
                                        (sql, connection)
            Dim reader As SqlDataReader
            reader = command.ExecuteReader()
            SqlContext.Pipe.Send(reader)
        End Using
    End Sub
End Class
```

ANALIZZIAMO IL CODICE

La prima cosa che notiamo è che la procedura è “Decorata” con un attributo “Microsoft.SqlServer.Server.SqlProcedure”. Tale attributo dice a SQL Server che la funzione FiltraUtenti è una Stored Procedure.
La procedura ha un parametro di tipo SqlString; il framework 2.0 fornisce, infatti, un insieme di tipi, inclusi nel namespace System.Data.SqlTypes, corrispondenti a quelli di SQL Server. Non è obbligatorio usarli ma è buona norma farlo.
Utilizziamo, quindi, un oggetto connection che inizializziamo passando al costruttore la stringa “context connection=true”. Stiamo utilizzando un managed provider per l’accesso ai dati chiamato SqlServer Data Provider ed implementato nella namespace System.Data.SqlServer. L’oggetto principale del namespace è SqlContext che rappresenta il contesto di esecuzione del managed code.
Abbiamo poi l’oggetto SqlPipe che è utilizzato per inviare i risultati dal server al client. Questo oggetto ha un metodo Send che ha quattro overload per consentire di inviare String, SqlError, SqlDataReader oppure SqlResultSet.
Con SqlContext.Pipe.Send(reader), utilizziamo il metodo Send per inviare un SqlDataReader ottenuto dall’esecuzione della query.
Una volta scritto il codice occorre fare i deploy della stored procedure: clicchiamo con il tasto destro del mouse sul nome del progetto e selezioniamo “Deploy”.

Fig. 4: **Deploy di un oggetto**

Abbiamo creato la nostra stored procedure che sarà visibile da SQL Server 2005. A questo punto, possiamo eseguirla da SQL Server.
Apriamo SQL Server Management Studio e colleghiamoci al nostro database

Fig. 5: **SQL Server Management Studio**

Creiamo una nuova query ed eseguiamo infine la Stored Procedure.

```
DECLARE @iniziale nvarchar(4000)
set @iniziale = 'a'
EXECUTE [Morpheusweb].[dbo].[FiltraUtenti]
  @iniziale
```

Se proviamo ad eseguire la query, invece della lista di utenti che ci aspettiamo, potremmo ottenere il seguente errore:

```
Execution of user code in the .NET Framework is
    disabled. Enable "clr enabled" configuration option.
```

Questo perché l’esecuzione di codice CLR è disabilitata di default su SQL Server. Per abilitarla, ci basta utilizzare una stored procedure di sistema sp_configure.

```
-- Abilitaiamo l’esecuzione di codice CLR
EXEC sp_configure 'clr enabled' , '1'
go
reconfigure;
```

A questo punto, potremo eseguire con successo la stored procedure.
Tramite Visual Studio 2005 è anche possibile eseguire il debug delle stored procedure e di tutti gli altri oggetti creabili in CLR. Per farlo, occorre inserire un breakpoint nel punto in cui vogliamo iniziare il debug e scrivere la query in cui utilizziamo l’oggetto in un apposito file, chiamato Test.sql, che Visual Studio crea automaticamente.

LE FUNZIONI

Tra gli oggetti di SQL Server che utilizziamo spesso ci sono anche le funzioni. Anche queste

possono essere scritte in CLR come le Stored Procedure.
Le funzioni sono molto simili a delle stored procedure, ma a differenza di queste devono restituire un valore.
Proviamo a realizzarne una che controlla, tramite una regular expression, se un indirizzo email è valido. Utilizzeremo poi la funzione come constraint per la tabella.
Creiamo una nuova funzione, vediamo il codice in C#

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.Text.RegularExpressions;
public partial class UserDefinedFunctions
{
  [SqlFunction(
    DataAccess=DataAccessKind.None,
    SystemDataAccess=SystemDataAccessKind
                                        None,
    IsDeterministic=true,
    IsPrecise=true)]
  public static SqlBoolean IsEmail(SqlString email)
  {
    string mail_regex = @"^([a-zA-Z0-9_\.
       \-])+\@(([a-zA-Z0-9\-]{2,})+\.)+([a-zA-Z0-
                                   9]{2,})+$";
    Regex expr = new Regex(mail_regex);
    return expr.IsMatch(email.Value);
  }
};
```

e il corrispondente in Visual Basic

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Data.SqlTypes
Imports Microsoft.SqlServer.Server
Imports System.Text.RegularExpressions

Partial Public Class UserDefinedFunctions
  <SqlFunction(DataAccess:=DataAccessKind.Read,
   SystemDataAccess:=SystemDataAccessKind.Read,
         IsDeterministic:=True, IsPrecise:=True)> _
  Public Shared Function IsEmail(ByVal email As
                            SqlString) As SqlBoolean
    Dim mail_regex As String = "^([a-zA-Z0-9_\.\-
         ])+\@(([a-zA-Z0-9\-]{2,})+\.)+([a-zA-Z0-
                                    9]{2,})+$"
    Dim expr As Regex = New Regex(mail_regex)
    Return expr.IsMatch(email.Value)
  End Function
End Class
```

NOTA

SQLTRIGGERCONTEXT

Fornisce informazioni sul trigger che viene scatenato. Ha una proprietà chiamata TriggerAction che ci dice quale azione ha scatenato il trigger (Insert, Update, Delete...)

Notiamo subito l'utilizzo dell'attributo SqlFunction (che indica una funzione di SQL Server) con alcune proprietà. DataAccess e SystemDataAccess impostate a None indicano che non viene utilizzato il Data Provider InProc di SQL Server mentre IsDeterministic è utilizzata per indicare che la funzione restituisce sempre lo stesso risultato a fronte di uno stesso valore in input.
Otteniamo il valore di ritorno della funzione tramite l'istruzione return. Una volta fatto il deploy, possiamo utilizzare la function come constraint sul campo email della nostra tabella utenti.

```
CREATE TABLE [dbo].[Utenti](
        [idUtente] [int] IDENTITY(1,1) NOT NULL,
        [nome] [varchar](50) COLLATE Latin1_
                      General_CI_AS NOT NULL,
        [cognome] [varchar](50) COLLATE
                Latin1_General_CI_AS NOT NULL,
        [email] [varchar](50) COLLATE Latin1_
                      General_CI_AS NOT NULL,
        [scadenza] [varchar](50) COLLATE
                Latin1_General_CI_AS NOT NULL
        CONSTRAINT [DF_Utenti_scadenza]
            DEFAULT (dateadd(year,(1),getdate())),
        CONSTRAINT [PK_Utenti] PRIMARY KEY
                                   CLUSTERED
        ( [idUtente] ASC) WITH (IGNORE_DUP_KEY
                      = OFF) ON  [PRIMARY]
                            ) ON [PRIMARY]
ALTER TABLE [dbo].[Utenti]
WITH CHECK ADD  CONSTRAINT [CK_Utenti_email]
CHECK  (([dbo].[isEmail]([email])=(1)))
```

Se proviamo ad eseguire una semplice query di inserimento

```
INSERT INTO [Morpheusweb].[dbo].[Utenti]
([nome] ,[cognome],[email])
VALUES ('Carmelo', 'Scuderi','mailerata')
```

Otteniamo un errore di SQL Server

```
"The INSERT statement conflicted with the CHECK
  constraint "CK_Utenti_email". The conflict occurred in
   database "Morpheusweb", table "dbo.Utenti", column
                                          'email'."
```

Utilizzando una mail corretta invece l'inserimento va a buon fine.

UTILIZZIAMO TRIGGERS

In modo simile a come visto per le stored procedure e le funzioni, è possibile creare dei managed trigger. Quando si crea un trigger occorre decorare la funzione con l'attributo SqlTrig-

ger che ha due parametri:

- Target: la tabella del nostro database su cui creare il trigger
- Event: l'evento che scatena il trigger (ad esempio: "FOR UPDATE", "FOR INSERT")

Per inviare mail tramite SQL Server in CLR, la proprietà "Permission Level" dell'assembly deve essere impostata a "Unsafe".

Fig. 8: ***Deploy di assembly Unsafe***

Quando facciamo il deploy potremmo ottenere un errore dovuto al fatto che SQL Server, per default, non consente il deploy di assemby che non siano Safe.
Per abilitare questa possibilità, occorre configurare il parametro "TRUSTWORTHY ON" sul database in cui dobbiamo creare il trigger. Per farlo eseguiamo la query:

```
ALTER DATABASE Morpheusweb SET TRUSTWORTHY
                                        ON;
```

Fatto questo, esaminiamo il codice del trigger, in C#

```
using System;
using System.Data;
using System.Data.SqlClient;
using Microsoft.SqlServer.Server;
using System.Data.Sql;
using System.Net.Mail;
public partial class Triggers
{
  [Microsoft.SqlServer.Server.SqlTrigger(
    Name="SendMail",
    Target="Utenti",
    Event="AFTER INSERT")]
  public static void SendMail()
  {
    SqlTriggerContext myContext = SqlContext.
                                  TriggerContext;
    if (myContext.TriggerAction == TriggerAction.
                                          Insert)
    {
      using (SqlConnection connection = new Sq
```

```
            Connection("context connection=true"))
      {
        connection.Open();
        string sql =
          "select nome, cognome, email, scadenza
                                    from inserted";
        SqlCommand command = new
                          SqlCommand(sql, connection);
        SqlDataReader myReader;
        using (myReader = command.
                                  ExecuteReader())
        {
          if (myReader.Read())
          {
            string nomeCompleto =
              myReader["nome"] + " " +
                                 myReader["cognome"];
            string email = myReader["email"].
                                        ToString();
            string scadenza = Convert.ToDate
                      Time(myReader["scadenza"])
                            .ToString("dd/MM/yyy");
            MailMessage myMail = new
                                    MailMessage();
            myMail.From = new MailAddress("web
                         master@morpheusweb.it");
            myMail.To.Add(new
                               MailAddress(email));
            myMail.Body =
              "Ciao " + nomeCompleto + ", grazie
                          per esserti iscritto\r\n" +
                         "La tua utenza scadrà il " +
                                        scadenza;
            myMail.IsBodyHtml = false;
            SmtpClient emailClient = new
              SmtpClient("IL_TUO_SERVER_SMTP");
            emailClient.Send(myMail);
          }
        }
      }
    }
  }
}
```

e il corrispondente in Visual Basic.NET

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Data.SqlTypes
Imports Microsoft.SqlServer.Server
Imports System.Net.Mail
Partial Public Class Triggers
 <Microsoft.SqlServer.Server.SqlTrigger(Name:="Send
  Mail", Target:="Utenti", Event:="AFTER INSERT")> _
   Public Shared Sub SendMail()
      Dim myContext As SqlTriggerContext =
                           SqlContext.TriggerContext
```

L'AUTORE

Carmelo Scuderi è ingegnere informatico. Si occupa di sviluppo software web-based per una società di telecomunicazioni di Milano. Gestisce un sito web ricco di script e manuali per chi si affaccia al mondo della programmazione web (www.morpheusweb.it)

SUL WEB

SQL Server 2005
http://www.microsoft.com/italy/sql/Default.mspx

SQL Server 2005 Express
http://msdn.microsoft.com/vstudio/express/sql/

SQL Server Central
http://www.sqlservercentral.com/

SqlJunkies
http://www.sqljunkies.com/

```
        If (myContext.TriggerAction = TriggerAction.
                                                   Insert) Then
            Using connection As New SqlConnection
                                   ("context connection=true")
                connection.Open()
                Dim sql As String = "select nome, cognome,
                                     email, scadenza from inserted"
                Dim command As SqlCommand = New S
                                  Command(sql, connection)
                Using myReader As SqlDataReader =
                                     command.ExecuteReader()
                    If (myReader.Read()) Then
                        Dim nomeCompleto As String =
                              myReader("nome").ToString() & _
                            " " + myReader("cognome").
                                                  ToString()
                        Dim email As String =
                                  myReader("email").ToString()
                        Dim scadenza As String = Convert.
                     ToDateTime(myReader("scadenza")).ToString
                                                ("dd/MM/yyy")
                        Dim myMail As MailMessage = New
                                                MailMessage()
                        myMail.From = New MailAddress
                                  ("webmaster@morpheusweb.it")
                        myMail.To.Add(New
                                          MailAddress(email))
                        myMail.Body = "Ciao " & nome
                                                  Completo & _
                          ", grazie per esserti iscritto\r\n" & _
                          "La tua utenza scadrà il " & scadenza
                        myMail.IsBodyHtml = False
                        Dim emailClient As SmtpClient = New
                           SmtpClient("IL_TUO_SERVER_SMTP")
                        emailClient.Send(myMail)
                    End If
                End Using
            End Using
        End If
    End Sub
End Class
```

SQL Server 2005 supporta l'oggetto SqlTriggerContext che viene utilizzato per ottenere informazioni sul trigger.
Tramite questo oggetto vediamo quale azione è stata intrapresa sulla tabella Utenti e, se si tratta di un inserimento (TriggerAction.Insert), eseguiamo il codice che invia la mail.
Per farlo, eseguiamo una query "select nome, cognome, email, scadenza from inserted" che recupera l'ultimo inserimento e creiamo una mail da inviare all'utente utilizzando l'oggetto MailMessagge del namespace System.Net.Mail.

Fig. 6: ***Debug di un oggetto***

T-SQL VS CLR

La possibilità di scrivere oggetti lato server utilizzando il framework .Net ci offre, un nuovo modo per sviluppare stored procedure, trigger o funzioni. Il rovescio della medaglia è che il programmatore deve decidere quando sviluppare in CLR e quando, invece, continuare ad utilizzare T-SQL.
In effetti, non c'è una risposta sempre valida ma delle linee guida. T-SQL va in genere utilizzato quando il nostro codice esegue prevalentemente accesso ai dati; il motore di esecuzione di SQL Server è molto complesso ed utilizza algoritmi avanzati per l'estrapolazione dei dati. CLR dovrebbe essere utilizzato quando le nostre procedure hanno una logica di programmazione molto complessa (ad esempio procedure ricorsive) o non possono essere espresse in T-SQL: con CLR otterremo prestazioni migliori e semplicità di implementazione.

CONCLUSIONI

Con l'integrazione del CLR 2.0 in SQL Server 2005 lo sviluppatore può implementare funzionalità molto complesse con estrema semplicità grazie alla possibilità di sfruttare i concetti di programmazione ad oggetti come l'ereditarietà e l'incapsulamento. È, inoltre, possibile sfruttare le numerose librerie messe a disposizione dal framework per compiti che non era possibile eseguire con le precedenti versioni di SQL Server.

Carmelo Scuderi

SQLPIPE

**L'oggetto SqlPipe consente di eseguire procedure direttamente sul database e poi inviare i risultati dal server al client. I due metodi principali dell'oggetto sono Send ed ExecuteAndSend.
ExecuteAndSend accetta come parametro un SqlCommand e come azione esegue la query passata come CommandText ed invia i risultati al client.
Send invia direttamente dei risultati al client. Ha quattro overload per consentire di inviare String, SqlError, SqlDataReader oppure SqlResultSet.**

OTTIMIZZAZIONE DI MYSQL

IL MODO IN CUI UNA QUERY VIENE SCRITTA PUÒ INFLUENZARE DECISAMENTE I TEMPI DI RISPOSTA DEL SERVER. MA QUALI SONO I PARAMETRI SU CUI BASARCI PER SCRIVERE QUERY EFFICIENTI? ECCO COME OTTENERE INFORMAZIONI UTILI PER OTTIMIZZARE LE INTERROGAZIONI

In questo articolo tenteremo di rispondere ad una domanda piuttosto frequente: "Perché il mio sito Web è lento?". Premesso che le risposte possono essere praticamente infinite, analizzeremo tutte quelle situazioni in cui la lentezza nella visualizzazione di una pagina Web dipende da una mancata ottimizzazione del database che contiene le informazioni utili a far girare la web application. Cosa succede ad esempio se dobbiamo recuperare da un database di 500.000 clienti le informazioni relative a un range di tutti i clienti che hanno effettuato un acquisto nel mese corrente?

Cinquecentomila è un numero abbastanza elevato da farci supporre che eseguire questa interrogazione sul database sia un'operazione abbastanza onerosa in termini di tempo. Il nostro scopo, in questo articolo, sarà appunto illustrare qualche tecnica per ottimizzare le prestazioni del Database. Nonostante questa premessa, dobbiamo dire, che la gamma delle possibili ottimizzazioni è talmente ampia che non possiamo dare una risposta certa ad ogni tipo di problema. La maggior parte degli utenti dovranno adattare le tecniche qui illustrate alla propria applicazione e spesso mescolarle fra loro fino a trovare la soluzione migliore. E' praticamente impossibile fornire una procedura standard che conduca alla "perfetta ottimizzazione" bisogna procedere per tentativi e "legare strettamente" il proprio problema alle tecniche illustrate.

REQUISITI

Conoscenze richieste
Principi di SQL

Software
Una qualunque versione di Linux o Windows. MySQL.

Impegno

Tempo di realizzazione

OTTIMIZZAZIONI LATO SOFTWARE

Nonostante il fatto che nessuna ottimizzazione è possibile se non eseguita in maniera congiunta dal sistemista e dallo sviluppatore, ci concentreremo su quello che maggiormente ci riguarda, ovvero come sviluppare codice che aumenti la velocità d'esecuzione delle query. Tralesceremo volutamente tutte le operazioni effettuabili sulla configurazione del server, quali ad esempio l'ottimizzazione della cache etc, ritenendo che questo genere di configurazione spetti più al sistemista che a noi.

LA SCELTA DEL FORMATO DELLE TABELLE

Alcuni di voi sapranno che MySQL offre diversi "engine" per la manipolazione delle tabelle sul server. Ora, nonostante la definizione rigorosa possa essere molto più ampia, per capire cosa stiamo cercando di fare è bene riportare il piano della discussione a un livello più basso. Una tabella contenente dei dati, si ridurrà per noi ad un file scritto in una particolare zona del disco. Il file conserverà i dati secondo un proprio schema, e ovviamente MySQL dovrà disporre di un meccanismo che gli consenta di prelevare i dati dal file in questione secondo la logica con cui sono stati conservati. Per cui quando parleremo di formati di tabelle, intenderemo il formato con cui le tabelle vengono scritte su disco, e quando parleremo di engine intenderemo il motore interno di mysql in grado di leggere e scrivere i dati nel file in questione con il formato dato. Per il nostro articolo analizzeremo quattro engine MyISAM, Heap, BDB, InnoDB. Ciascuno di questi ha particolarità proprie. Prima di procedere però alla descrizione delle caratteristiche dei vari engine sarà bene dare corpo a qualche definizione che altrimenti potrebbe sembrare oscura.

IL PROBLEMA DEL LOCK

Supponiamo che i nostri dati siano contenuti in un unico file. Ad un certo momento l'utente A sta cercando di scrivere un dato, contemporaneamente l'utente B sta facendo la stessa cosa. Il risultato è che se tutti e due scrivono contemporaneamente il file potrebbe risultare danneggiato o inconsistente. Per cui normalmente il processo che si esegue è quello di assegnare in modo univoco la risorsa a chi la detiene. Ad esempio l'intero file viene assegnato all'utente A e viene messo in stato di "lock", ovvero nessuno può scrivere sul file fino a che l'utente A non ha terminato le sue operazioni. Quando l'utente A termina, il file viene sbloccato, assegnato alla risorsa B, e messo nuovamente in stato di lock e così via. Apparentemente è una soluzione semplice, tut-

tavia se le operazioni di scrittura sono frequenti inevitabilmente ci sarà una coda di attesa lunghissima, pertanto un dato verrà realmente scritto nel file con tempi di ritardo notevoli rispetto a quando l'utente l'ha inserito, con tutti i problemi che ne possono derivare.
Apparentemente il problema l'utente A legge l'utente B scrive è molto più semplice. Se l'utente A sta leggendo un dato e l'utente B ne sta scrivendo un altro, niente di male può accadere. Ma se per pura sfortuna l'utente B sta eliminado il dato letto da A si possono riverificare problemi di consistenza. Qui entrano in gioco i così detti shared locks e exclusive locks. Ovvero se l'utente A tenta di leggere un file viene applicato un lock di tipo shared, ovvero molti client possono continuare ad accedere in lettura, se invece un utente tenta di scrivere ottiene un exclusive lock ovvero sarà lui il proprietario della risorsa e nessun altro potrà leggere o scrivere fino a quando non la rilascia. Uno dei compiti principali degli engine di MySQL è dunque la gestione dei lock. L'engine in modo del tutto trasparente all'utente deve continuamente stabilire su cosa effettuare un lock e con che modalità. I vari tipi di lock offerti dai vari engine rappresentano una delle basi più importanti per capire quale formato di tabelle adottare nella nostra applicazione.

LOCK SU TABELLE

Nelle prime versione di MySQL l'engine di default era MyISAM. In questo formato l'unico metodo di lock utilizzato è il lock sull'intera tabella. Questo significa che quando un utente tenta di scrivere, l'intera tabella viene bloccata e nessun altro riuscirà ad utilizzarla in lettura o scrittura. Questo apparentemente è un grave limite. Tuttavia MySQL nel corso degli anni si è dimostrato velocissimo pure utilizzando questo tipo di lock nel suo engine di default. Il motivo di questo è da ricercarsi nel fatto che le applicazioni servite da MySQL sono prevalentemente Web Application e pertanto il più delle volte si verificano centinaia se non migliaia di operazioni di lettura contemporanee, mentre le operazioni di scrittura risultano piuttosto ridotte. Per tale motivo aumentare la complessità dell'engine aggiungendo una gestione dei lock più granulare avrebbe diminuito la velocità dell'algoritmo senza portare vantaggi notevoli all'applicazione

LOCK SU PAGINE

Se la vostra applicazione utilizza un grado di concorrenza moderato nella scrittura/lettura delle tabelle, una buona soluzione è utilizzare l'engine Berkeley DB. Questo tipo di Engine blocca in scrittura/lettura una porzione della tabella, ovvero quella interessata dall'operazione di scrittura. Questa porzione della tabella prende il nome di "pagina" e per Berkeley DB ogni "pagina" ha una dimensione di 8Kb.

LOCK SU RIGHE

InnoDB offre il maggior grado di concorrenza. Con questo engine il lock avviene sulla singola riga interessata dall'operazione di scrittura. Oltre a questo tipo di protezione InnoDB utilizza un meccanismo noto come MVCC ovvero Multi Version Concurrency Control. In sostanza l'engine mantiene per ogni riga due informazioni addizionali nascoste al programmatore ovvero l'ID_CREAZIONE, l'ID_CANCELLAZIONE. Rispettivamente questi due campi contengono rispettivamente il momento in cui la riga e stata creata e il momento in cui la riga è stata cancellata. Oltre a questo, ad ogni transazione eseguita, il database aumenta di un'unità un numero di versione del database. A questo punto entra in gioco un meccanismo di confronto che fa si che solo le righe che soddisfino i criteri di consistenza del database vengano restituite all'utente. Tutto questo meccanismo ovviamente fa si che InnoDB debba mantenere costantemente aggiornate le informazioni di versioning delle righe e questo aumenta esponenzialmente la complessità delle operazioni. Per tanto InnoDB è da usarsi soltanto se davvero avete bisogno di una gestione delle concorrenza delle operazioni veramente alta e sicura.

GLI INDICI

Uno dei concetti più importanti in tema di ottimizzazione di database è quello degli indici. Supponete di avere una tabella contenente 500.000 prodotti. Da questa tabella volete ricavare l'unico prodotto corrispondente al nome "Mastice AttaccaTutto in un battibaleno". Tecnicamente l'engine dovrebbe scorrere l'intera tabella dal primo elemento fino all'ultimo, confrontare il contenuto del campo "nome_prodotto" con quello dato ed eventualmente restituire un risultato se la condizione fosse verificata. Se non avessimo a disposizione un computer ma un catalogo cartaceo tutto sarebbe estremamente più semplice. Molto probabilmente il catalogo sarebbe ordinato in ordine alfabetico per nome_prodotto, sarebbe sufficiente dunque portarsi alla lettera "M" e pescare il prodotto desiderato seguendo l'ordine alfabetico. L'uso degli indici all'interno di un database risponde esattamente alle stesse esigenze. Sostanzialmente un indice è una tabella di valori ordinati che puntano a dati. Va da se che l'indicizzazione di una tabella occupa spazio, e allo stesso

COSA SONO LE TRANSAZIONI?

Per transazione si intende un gruppo di istruzioni SQL che viene gestito in maniera atomica, cioè come se fosse un'unica operazione. Usare le transazioni garantisce un lock esclusivo su tutte le operazioni che coinvolgono la transazione. In MySQL gli engine che supportano le transazioni sono Berkeley DB e InnoDB. MyISAM non gestisce le transazioni! Ovviamente il prezzo se si usano le transazioni è una maggiore complessità della gestione e di conseguenza in molti casi si può verificare una diminuzione della velocità. Ovviamente tutto dipende da cosa dovete ottenere. Se la vostra applicazione non deve gestire molte operazioni di scrittura contemporanee e non ha bisogno di un controllo fine sul lock, l'engine migliora da utilizzare rimane MyISAM.

modo appare evidente che mantenere costantemente ordinato un indice implica molte operazioni di update sull'indice stesso. Ovvero ogni volta che la tabella viene modificata, l'indice corrispondente deve essere riordinato sulla base delle modifiche effettuate. Perciò nell'utilizzare gli indici bisogna tener conto dell'occupazione dello spazio e del carico a cui si sottopone la CPU per la gestione dell'indice stesso. Creare un indice in MySQL si riduce in buona parte alla seguente istruzione:

```
ALTER Table NomeTabella ADD Index(nomecampo);
```

Ad esempio l'istruzione

```
ALTER TABLE Prodotti ADD Index(nome_prodotto);
```

Crea un indice della tabella Prodotti ordinato secondo il campo nome_prodotto. Poiché abbiamo detto che va trovato sempre il massimo bilanciamento fra la dimensione dell'indice e le risorse a disposizione è anche possibile creare degli indici parziali, ad esempio l'istruzione

```
ALTER TABLE Prodotti ADD Index(nome_prodotto(3));
```

Indicizza i prodotti tenendo conto solo delle prime 3 lettere del nome del prodotto, così che la query

```
Select * from prodotti WHERE
                    nome_prodotto="Mastice";
```

restituisce Mastice ma anche Massaggiatore, proprio perché la query viene effettuata tenendo conto dei primi tre valori del nome_prodotto. Allo stesso modo è possibile creare indici multicolonna con la seguente sintassi

```
ALTER TABLE Prodotti ADD Index(nome_prodotto,
                    prezzo_prodotto);
```

In questo modo si produce un indice che ordina i dati prima secondo il nome del prodotto e poi secondo il prezzo. Come visto in precedenza lo stesso indice può essere creato in modo parziale

```
ALTER TABLE Prodotti ADD Index(nome_prodotto(3),
                    prezzo_prodotto(3));
```

Attenzione che la creazione di un indice multicolonna non corrisponde alla creazione di due indici separati. MySQL è in grado di utilizzare un solo indice per volta! La creazione di un indice multicolonna consente di elaborare un solo elenco che utilizza due criteri di ordinamento. E' sempre possibile creare due indici:

```
ALTER TABLE Prodotti ADD Index(nome_prodotto);
ALTER TABLE Prodotti ADD Index(prezzo_prodotto);
```

Ma in tal caso quando si esegue la query MySQL sceglierebbe l'indice più conveniente per esaudire rapidamente la richiesta. Consideriamo la seguente query:

```
Select * from Prodotti where nome_prodotto='Mastice'
                    AND prezzo ='10';
```

Con la creazione di un unico indice multicolonna MySQL selezionerebbe prima un insieme di prodotti che soddisfano alla prima condizione, ma parallelamente sarebbe in grado di utilizzare un albero di ricerca per soddisfare la seconda condizione, per cui la velocità aumenterebbe notevolmente. Nel secondo caso invece potrebbe utilizzare un solo indice per volta e la velocità diminuirebbe.

LA PAROLA CHIAVE "EXPLAIN"

Avendo fornito alcuni concetti base sulle tabelle e sugli indici, possiamo passare ad analizzare le nostre query, nella speranza di capire come ottimizzarle. Prima di tutto, per ottenere informazioni su come è composta una nostra tabella possiamo utilizzare la parola chiave Describe, ad esempio:

```
mysql> Describe msvcs_comuni;
```

Come si vede è una tabella che contiene tutti i comuni d'Italia. Non abbiamo definito ancora alcun indice ne effettuato particolari ottimizzazioni. Proviamo a fare un test su una nostra query utilizzando la parola chiave explain:

```
mysql> explain select Comune,ComuneSigla,ComuneCAP
        from msvcs_comuni where Comune='Roma' \G
id: 1
select_type: SIMPLE
table: msvcs_comuni
type: ALL
possible_keys: NULL
```

QUALE ENGINE SCEGLIERE

Come già detto tutto dipende da cosa volete realizzare. Se la vostra tabella non coinvolge molte operazione di scrittura contemporanee e non ha bisogno di gestire le transazioni, la scelta migliore rimane MyISAM. Ricordatevi anche che la scelta dell'engine non coinvolge l'intero database ma solo le singole tabelle. Pertanto potete sempre decidere di utilizzare un formato per una certa tabella e un altro per una tabella che deve assolvere a funzioni diverse.

NOTA

TABELLE HEAP

Le tabelle HEAP sono particolari strutture dati che non vengono scritte su Disco ma che mantengono le informazioni direttamente in memoria. Inutile dire che questo genere di tabelle sono velocissime. Di contro le tabelle Heap non gestiscono completamente gli indici in modo ottimale e ovviamente consumano quantità di memoria gigantesche

Field	Type	Null	Key	Default	Extra
comuneID	int(11)		PRI	NULL	auto_increment
Comune	varchar(30)		MUL		
comuneSigla	varchar(4)	YES		NULL	
comunePrefTel	varchar(5)		MUL		
comuneCAP	varchar(5)		MUL		
REF_MAP	varchar(100)				
map	varchar(255)				

```
key: NULL
key_len: NULL
ref: NULL
rows: 8208
Extra: Using where
1 row in set (0.00 sec)
```

Cerchiamo di capire cosa vuol dire questa risposta.

id => contiene un identificatore numerico di ogni tabella coinvolta nell'interrogazione
select_type => identifica un "ruolo" per la tabella interrogata all'interno dell'interrogazione. Chiariremo in un secondo momento cosa vuol dire esattamente "ruolo"
table => contiene il nome della tabella interrogata
type => contiene il tipo di join utilizzato. Chiariremo anche questo punto in un secondo momento
possible_keys=> contiene un elenco di possili key che il server può utilizzare. Questo parametro sarà piuttosto importante per l'ottimizzazione degli indici. Se la tabella non è dotata di nessun indice, in questo campo troveremo il valore "Null"
key => contiene il valore dell'indice che mysql ha deciso di utilizzare
key_len => contiene il valore in byte della dimensione della chiave
ref => contiene le colonne che sono state utilizzate per confronto con la chiave
rows => contiene il numero di chiavi che MySQL deve esaminare per soddisfare l'interrogazione
Extra => contiene informazioni aggiuntive che possono esserci utili in un secondo momento

UNA PRIMA OTTIMIZZAZIONE

Concentriamoci prima di tutto sul valore contenuto in rows. Per la nostra prima query la risposta era la bellezza di 8208. Non esistendo un indice, MySQL dovrà scansionare tutte le 8208 righe della tabella e confrontarle con la costante che gli abbiamo fornito come parametro. Proviamo ad aggiungere un indice per vedere come varia il comportamento di MySQL:

```
mysql> alter table msvcs_comuni add index(Comune);
Query OK, 8208 rows affected (0.46 sec)
Records: 8208  Duplicates: 0  Warnings: 0
```

e rieseguiamo la query con l'explain

```
mysql> explain select Comune,ComuneSigla,ComuneCAP
        from msvcs_comuni where Comune='Roma' \G
id: 1
select_type: SIMPLE
table: msvcs_comuni
type: ref
possible_keys: Comune
key: Comune
key_len: 30
ref: const
rows: 1
Extra: Using where
1 row in set (0.00 sec)
```

Questa volta in rows notiamo il valore 1. Avendo aggiunto un indice corretto le righe interessate si riducono da 8208 ad appena 1! All'interno della stessa risposta notiamo che Type è passato da All a Ref. Allo stesso modo possible_keys contiene l'elenco delle possibili chiavi utilizzabili, in questo caso solo "Comune" e infine che il server ha deciso di utilizzare proprio la chiave "Comune". Ora, concentriamoci sul "Type", la nostra explain indica che il tipo di Join è un "Ref". Esattamente cosa vuol dire?
In pratica "Comune" non dispone di un "indice a chiave unica" ovvero potrebbero esistere più comuni con lo stesso nome. Per questo motivo MySQL è costretto ad usare una query incrociata fra i valori ritornati dall'indice e la tabella stessa. Proviamo a trasformare "Comune" in una chiave unica. Ovvero non ci possono essere valori ripetuti per il campo "Comune" all'interno della tabella

```
ALTER TABLE `msvcs_comuni`
ADD UNIQUE (`Comune`)
```

e rieseguiamo l'explain

```
mysql> explain select Comune,ComuneSigla,ComuneCAP
        from msvcs_comuni where Comune='Roma' \G
id: 1
select_type: SIMPLE
table: msvcs_comuni
type: const
possible_keys: Comune_2,Comune
key: Comune_2
key_len: 30
```

```
ref: const
rows: 1
Extra:
1 row in set (0.00 sec)
```

Questa volta il type è diventato di tipo "Const" ovvero la tabella viene aperta una volta sola, siamo certi che c'è un solo valore che soddisfa la richiesta. Le ricerche di tipo Const sono velocissime!
Notiamo però che è stato creato automaticamente un indice Comune_2 che questa volta è di tipo "a chiave univoca" e che MySQL ha scelto di utilizzare questo tipo di indice piuttosto che il precedente.

RICHIESTE DI INTERVALLI

Proviamo adesso un'interrogazione che coinvolge un certo numero di dati:

```
mysql> explain select Comune,ComuneSigla,ComuneCAP
    from msvcs_comuni where Comune between 'Ragusa'
                                       and 'Roma' \G
id: 1
select_type: SIMPLE
table: msvcs_comuni
type: range
possible_keys: Comune_2,Comune
key: Comune
key_len: 30
ref: NULL
rows: 282
Extra: Using where
1 row in set (0.01 sec)
```

Notiamo che adesso il type è diventato "Range" e che il valore restituito in rows è 282. Cioè MySQL pensa che ci possano essere 282 record fra Ragusa e Roma, tuttavia se eseguiamo realmente la query ci accorgeremo che vengono restituiti appena 273 record. Questo perché MySQL effettua una stima di quante righe possano essere coinvolte nel confronto. In questo caso utilizzando l'indice "Comune" vengono confrontate 282 righe delle quali 273 soddisfano la condizione.

RICERCHE SU STRINGHE

Ora proviamo la seguente query:

```
mysql> explain select Comune,ComuneSigla,ComuneCAP
    from msvcs_comuni where Comune like 'Ro%' \G
id: 1
select_type: SIMPLE
table: msvcs_comuni
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 8200
Extra: Using where
1 row in set (0.01 sec)
```

Notate che non viene utilizzato nessun indice, e che le righe interrogate tornano ad essere 8200. Cosa è successo? L'operatore Like non fa uso degli indici, ovvero utilizza un particolare algoritmo interno per cui l'indicizzazione non risulta efficiente. In caso di tabelle di grandi dimensioni è utile creare un indice 'full text' e modificare la query utilizzando l'operatore 'match'. Vediamo come:

```
ALTER TABLE `msvcs_comuni` ADD
                              FULLTEXT(`Comune`)
```

In questo modo creiamo l'indice full text. Proviamo ora a riscrivere la query utilizzando l'operatore match:

```
mysql> explain select Comune,ComuneSigla,ComuneCAP
    from msvcs_comuni where match(comune) against
                       ('Ro*' IN BOOLEAN MODE) \G
id: 1
select_type: SIMPLE
table: msvcs_comuni
type: fulltext
possible_keys: Comune_3
key: Comune_3
key_len: 0
ref:
rows: 1
Extra: Using where
1 row in set (0.01 sec)
```

Notate che adesso il type è diventato di "fulltext" e che viene utilizzato l'indice Comune_3 ovvero quello creato appositamente per ricerche di questo tipo. Anche in questo caso c'è da annotare che gli indici full text occupano spazio e come tali è necessario utilizzarlo quando strettamente necessario

CONCLUSIONI

Abbiamo appena sfiorato la punta dell'iceberg delle tecniche di ottimizzazione in MySQL. In particolar modo l'uso sapiente dell'istruzione Explain e degli indici consente di ottenere miglioramenti significativi nel recupero dei dati. Noterete che non abbiamo approfondito alcune tematiche relative agli indici o alle chiavi, il nostro intento era fornire le basi per iniziare a capire come lavora MySQL. In futuro non mancheremo di continuare ad occuparci dell'argomento andando sempre più in profondità

FRAMMENTI DI SAGGEZZA

CAPITA SPESSO DI VOLER RIUTILIZZARE UNO SPEZZONE DI CODICE PRODOTTO IN PRECEDENZA, ALL'INTERNO DI UN NUOVO PROGETTO. DOVE CERCARLO? CON GLI "SNIPPET" DI VISUAL STUDIO È POSSIBILE MANTENERE UN ARCHIVIO MOLTO PARTICOLARE...

"Dove l'avevo messo?" - Questa è la "fatidica" domanda che ogni programmatore si pone un giorno sì e l'altro ... pure. A chiunque di noi con un minimo di anni di esperienza sulle spalle è capitato di affrontare e risolvere con successo dei problemi specifici: aprire una connessione a un particolare database, impostare un controllo in una certa maniera ecc...

Il problema è che, quando ci troviamo davanti allo stesso problema, ci ricordiamo sì, vagamente, che la cosa "non ci è nuova", che da qualche parte dovremmo aver già scritto il codice che ci tornerebbe proprio comodo, ma alla fine, spesso, ci rassegniamo e cominciamo tutto daccapo.

È per far fronte a questo problema che Microsoft ha inserito in Visual Studio 2005 (anche nelle versioni Express) un utile strumento chiamato frammenti di codice o, molto più simpaticamente, in inglese snippets.

COSA SONO GLI SNIPPETS

Per capire cosa sono gli snippets la cosa migliore è provarli.

Poniamoci quindi un obiettivo "reale" diciamo che il nostro scopo sarà quello di costruire una classe, in Visual Basic , che ha il compito di criptare e decriptare un file di testo.

REQUISITI

Conoscenze richieste
conoscenza di base di VB.NET

Software
Visual Studio 2005 anche in versione Express

Impegno

Tempo di realizzazione

Fig. 1: Aggiunta di una classe al progetto

Per prima cosa, in un progetto di Visual Studio o di Visual Studio Express, aggiungiamo una nuova classe, come in figura 1.

Impostiamo quindi un metodo vuoto che compie l'azione di criptazione del file:

```
Public Sub EncryptFile(ByVal filename As String,
                       ByVal textToEncrypt As String)
End Sub
```

All'interno del corpo del metodo (nella riga vuota) facciamo clic con il tasto destro del mouse in modo di visualizzare il pop-up come in Figura 2 :

Fig. 2: Inserimento del frammento di codice

Al primo posto, tra le voci di menu, troviamo proprio quella che fa al caso nostro : "Inserisci frammento di codice". Se clicchiamo su questa voce apparirà una struttura a cartelle come quella mostrata in Figura 3.

Fig. 3: Libreria di frammenti di codice per argomento

Di qui scegliamo la cartella Protezione (in inglese Security) e l'apriamo con doppio clic per mostrare i frammenti che contiene, come vediamo in Figura 4.

Fig. 4: ***Frammenti di codice per l'argomento selezionato***

Qui scegliamo il frammento (snippet) dal titolo "Esegui la crittografia di una stringa". Notiamo anche che lo snippet ha un Tip che ci informa sul contenuto del codice e sulla possibilità di utilizzare un tasto di scelta rapida per l'operazione di inserimento. Questo vuol dire che possiamo inserire direttamente il frammento di codice direttamente scrivendo il nome di scelta rapida (in questo caso secEncrypt) e premendo il tasto TAB.
Una volta cliccato il nome corrispondente al frammento di codice verrà inserito il codice corrispondente nel punto dell'Editor dov'è posizionato il cursore, Figura 5.

Fig. 5: ***Il frammento di codice inserito***

Notiamo che ci sono delle parti di codice evidenziate in verde che sono quelle che devono essere adattate al caso corrente e che, in testa al codice, vengono inseriti automaticamente gli imports necessari per gli spazi di nomi utilizzati. Questo non è un articolo sulla crittografia, per cui non ci soffermiamo a descrivere tutti i passaggi del codice. Ci limitiamo a dire che, nella nostra classe, è necessario implementare un metodo che trasformi una password in una chiave e un vettore (matrici di byte) validi per l'algoritmo utilizzato.
Nel nostro caso abbiamo fatto così:

```
Private _Password As String = "password"
Public Property Password() As String
    Get
        Return _Password
    End Get
    Set(ByVal Value As String)
        _Password = Value
    End Set
End Property
Private ReadOnly Property Key() As Byte()
    Get
```

```
        Return
System.Text.Encoding.UTF8.GetBytes(Password.Pa
                                dRight(32, "="))
    End Get
End Property

Private ReadOnly Property IV() As Byte()
    Get
        Return
System.Text.Encoding.UTF8.GetBytes(Password.Pa
                                dRight(16, "="))
    End Get
End Property
```

Quindi andiamo a modificare il codice inserito dallo snippet inserendo i valori corretti nei segnaposti e otteniamo il nostro metodo:

```
Public Sub EncryptFile(ByVal filename As String,
                ByVal textToEncrypt As String)
    Dim fStream As FileStream =
    File.Open(filename, FileMode.OpenOrCreate)
    Dim RijndaelAlg As Rijndael =
                            Rijndael.Create
    Dim cStream As NewCryptoStream(fStream,
                                            _
RijndaelAlg.CreateEncryptor(Me.Key, Me.IV), _
CryptoStreamMode.Write)
    Dim sWriter As New StreamWriter(cStream)
    sWriter.WriteLine(textToEncrypt)
    sWriter.Close()
    cStream.Close()
    fStream.Close()
End Sub
```

Per la decriptazione di un file codificato in una stringa, operazione inversa, impostiamo, anche qui, una funzione vuota:

```
Public Function DecryptFile(ByVal filename As
                        String) As String
End Function
```

Anche per questa operazione possiamo utilizzare uno snippet già pronto: "Esegui la decrittografia di un testo da un file" da inserire nel corpo della funzione che, opportunamente adattata, si presenterà così:

```
Public Function DecryptFile(ByVal filename As
                        String) As String
    Dim fStream As FileStream =
    File.Open(filename, FileMode.OpenOrCreate)
    Dim RijndaelAlg As Rijndael =
                            Rijndael.Create
    Dim cStream As New
                    CryptoStream(fStream, _
    RijndaelAlg.CreateDecryptor(Me.Key, Me.IV), _
```

RISORSE SUL WEB

Un buon articolo, per iniziare, è pubblicato in http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvs05/html/codesnippets.asp
In genere http://msdn.microsoft.com/library rappresenta la reference principale per questa tecnologia.

```
                                  CryptoStreamMode.Read)
        Dim sReader As New
                                StreamReader(cStream)
        Dim plainText As String =
                                      sReader.ReadLine()
        sReader.Close()
        cStream.Close()
        fStream.Close()
        Return plainText
    End Function
```

ANATOMIA DI UNO SNIPPET

Fin qui abbiamo visto come utilizzare gli snippets offerti dall'ambiente di Visual Studio (che già di per sé sono un buon numero).
Il vero punto di forza di questa tecnologia sta però nel fatto di poter scrivere dei propri snippet con il codice che utilizziamo più frequentemente. Prendiamo, ad esempio, la classe che abbiamo visto prima, per la criptazione e decriptazione di testo in un file. Per le operazioni principali siamo stati aiutati dagli snippet predefiniti, tuttavia alcuni passi del codice li abbiamo dovuti inserire a mano: in particolare la trasformazione di una password di testo in una matrice di Byte. Quest'azione potrebbe essere utile anche altre volte, quindi potrebbe essere un buon candidato per un nuovo snippet. Visual Studio offre uno strumento per gestire gli snippets, si trova sotto il menu strumenti alla voce "Gestione frammenti di codice", Figura 6.

Fig. 6: **Gestione frammenti di codice**

La finestra di gestione relativa è molto semplice, Figura 7, e consente di impostare, aggiungere o rimuovere elementi o gruppi di elementi.Notiamo che, selezionando uno sni pet, appare anche il relativo percorso del file su disco con estensione .snippet .
Se andiamo a visualizzare il contenuto della directory a cui appartiene il file vediamo che, in questo caso, il file dello snippet è insieme a tutti gli altri concernenti lo stesso argomento. Se apriamo un file, ad esempio quello della crittografia di una stringa, notiamo che lo

Fig. 7: **La finestra di gestione degli snippet**

stesso è, in realtà, in formato XML:

```
<?xml version="1.0"?>
<CodeSnippets
xmlns="http://schemas.microsoft.com/VisualStudi
                              o/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title>Esegui la crittografia di una
                                    stringa</Title>
      <Author>Microsoft Corporation</Author>
      <Description>Esegue la crittografia di un
           testo da un file utilizzando l'algoritmo di
                  crittografia Rijndael.</Description>
      <Shortcut>secEncrypt</Shortcut>
    </Header>
    <Snippet>
      <References>
        <Reference>
<Assembly>System.Security.dll</Assembly>
        </Reference>
      </References>
      <Imports>
        <Import>
<Namespace>Microsoft.VisualBasic</Namespace>
        </Import>
        <Import>
          <Namespace>System.IO</Namespace>
        </Import>
        <Import>
<Namespace>System.Security.Cryptography</N
                                        mespace>
        </Import>
      </Imports>
      <Declarations>
        <Literal>
          <ID>fileName</ID>
          <Type>String</Type>
          <ToolTip>Il nome del file contenente il
                        testo crittografato.</ToolTip>
          <Default>"encrypted.txt"</Default>
        </Literal>
        <Object>
```

```
        <ID>privateKey</ID>
        <Type>Byte</Type>
        <ToolTip>La chiave privata utilizzata per
                 crittografare i dati.</ToolTip>
        <Default>RijndaelAlg.Key</Default>
      </Object>
      <Object>
        <ID>initializationVector</ID>
        <Type>Byte</Type>
        <ToolTip>Il vettore di inizializzazione
    utilizzato nello schema di crittografia con chiave
                                     simmetrica.
</ToolTip>
        <Default>RijndaelAlg.IV</Default>
      </Object>
      <Literal>
        <ID>plainText</ID>
        <Type>String</Type>
        <ToolTip>La variabile di tipo stringa
    contenente il testo da crittografare.</ToolTip>
        <Default>"Text to encrypt"</Default>
      </Literal>
    </Declarations>
    <Code Language="VB" Kind="method
  body"><![CDATA[Dim fStream As FileStream =
   File.Open($filename$, FileMode.OpenOrCreate)
      Dim RijndaelAlg As Rijndael =
                                  Rijndael.Create
      Dim cStream As New
                            CryptoStream(fStream,
         RijndaelAlg.CreateEncryptor($privatekey$,
                        $initializationVector$), _
     CryptoStreamMode.Write)
      Dim sWriter As New StreamWriter(cStream)
      sWriter.WriteLine($plainText$)
      sWriter.Close()
      cStream.Close()
      fStream.Close()]]></Code>
  </Snippet>
 </CodeSnippet>
</CodeSnippets>
```

La sintassi è molto semplice (tra l'altro Visual Studio offre anche il supporto di intellisense per la modifica degli snippets). L'elemento CodeSnippets è quello di primo livello, e contiene uno o più elementi CodeSnippet che costituisce l'elemento base per definire il nostro frammento di codice.
CodeSnippet si articola in due sotto-elementi:
1. Header che contiene le informazioni descrittive del frammento
2. Snippet che contiene il codice e gli altri elementi utili a definirne il contesto

L'ELEMENTO HEADER

In Header possono essere contenuti gli elementi:

Nome elemento	Tipo	Descrizione	Min./Max
Author		facoltativoNome dell'autore	0/1
Description	facoltativo	Descrizione del frammento che appare nel Tooltip	0/1
HelpURL	facoltativo	URL Contenente ulteriori informazioni	0/1
Keywords	facoltativo	Contiene elementi Keyword, ovvero parole chiave personalizzate a scopo di ricerca o categorizzazione	0/1
Shortcut	facoltativo	Testo di collegamento utilizzabile per inserire il frammento con l'operazione: <testo> + TAB	0/1
SnippetTypes	facoltativo	Contiene elementi SnippetType che specificano il comportamento dello snippet	0/1
Title	obbligatorio	Nome dello snippet come appare in Intellisense	1/1

Tabella 1: Gli elementi all'interno del file XML

È da notare qui l'elemento SnippetTypes che contiene elementi di tipo SnippetType. I valori di un elemento SnippetType possono essere:

- SurroundsWith: permette di inserire il frammento di codice intorno a un segmento di codice.
- Expansion: permette di inserire il frammento di codice nella posizione del cursore.

L'ELEMENTO SNIPPET

L'elemento snippet è più articolato e permette di definire, oltre al codice vero e proprio, il contesto dove esso si colloca.
Qui possiamo avere i gruppi di elementi:

- eferences
- Import
- Declarations
- Code

L'elemento References definisce i riferimenti (a librerie di sistema o personali) che verranno aggiunti automaticamente al progetto (se non già presenti) al momento in cui si inserisce il frammento.
Tali riferimenti sono definiti in un numero illimitato di sotto-elementi Reference che contengono: un elemento Assembly per definire il nome dell'assembly che verrà aggiunto al progetto e, opzionalmente, un elemento Url contenente l'indirizzo web dove è possibile trovare informazioni sull'assembly aggiunto.
Ad esempio:

```
<References>
        <Reference>
<Assembly>System.Security.dll</Assembly>
```

```
        <Url>http://www.microsoft.com</Url>
      </Reference>
</References>
```

L’elemento Imports , invece, definisce gli spazi dei nomi che vengono automaticamente aggiunti in testa al file di codice utilizzando lo snippet, questo evita di dover far riferimento al nome completo dei tipi nel codice che andremo a definire. Imports contiene più elementi Import che, sotto l’elemento Namespace, definiscono lo spazio dei nomi relativo. Ad esempio:

```
<Imports>
      <Import>
<Namespace>Microsoft.VisualBasic</Namespace>
      </Import>
</Imports>
```

L’elemento Declarations è invece importante perché definisce i valori letterali o gli oggetti del frammento di codice che l’utente può modificare (quelli che vengono evidenziati in verde). Segnaposto dei valori effettivi.
Tali valori sono espressi da due tipi di elementi: Literal per i valori letterali e Object per gli oggetti. Ad esempio:

```
<Declarations>
      <Literal>
        <ID>fileName</ID>
        <ToolTip>Il nome del file contenente il
                   testo crittografato.</ToolTip>
        <Default>"encrypted.txt"</Default>
      </Literal>
      <Object>
        <ID>privateKey</ID>
        <Type>Byte</Type>
        <ToolTip>La chiave privata utilizzata per
                   crittografare i dati.</ToolTip>
        <Default>RijndaelAlg.Key</Default>
      </Object>
...
```

Ogni elemento Literal o Object deve definire un elemento ID che possa essere richiamato nel codice e un elemento Default che contiene il valore predefinito. È possibile anche definire un elemento ToolTip che verrà mostrato quando l’utente si sposta sul segnaposto per descrivere l’oggetto o il valore. Per gli elementi Object è obbligatorio anche il sotto-elemento Type che definisce il tipo a cui appartiene l’oggetto. All’interno del codice possiamo inserire un segnaposto definito come Literal o Object attraverso il suo ID racchiuso tra il simbolo $ (o altro simbolo che possiamo definire noi nell’elemento Code, vedi successivamente). Ad esempio : $fileName$.

L’elemento Code è, naturalmente quello più importante, perché racchiude il nostro codice. Tale elemento ha tre attributi:

- Delimiter (facoltativo) - specifica il delimitatore utilizzato per descrivere i valori letterali e gli oggetti nel codice. Se non altrimenti definito, il delimitatore è $.
- Kind (facoltativo) - specifica il tipo di codice contenuto nel frammento e pertanto le posizioni in cui il frammento può essere inserito. I valori disponibili sono;
- method body – il frammento deve essere inserito all'interno di una dichiarazione di metodo
- method decl - il frammento deve essere inserito all'interno di una classe o un modulo
- type decl - il frammento deve essere inserito all'interno di una classe, un modulo o uno spazio dei nomi
- page - il frammento di codice deve essere utilizzato come codice all'interno di una pagina di progetto Web
- file - il frammento è un file di codice completo
- any - il frammento può essere inserito in qualsiasi posizione
- Language (obbligatorio) - specifica il linguaggio del frammento di codice. I valori disponibili sono;
  - VB
  - CSharp
  - VJSharp
  - XML

All’interno dell’elemento Code viene inserito il codice vero e proprio, preferibilmente all’interno dei tag <![CDATA[...]]> per evitare che alcuni simboli presenti nel codice (ad esempio < o >) possano essere interpretati come XML.

IL NOSTRO SNIPPET

A questo punto possiamo procedere a creare il nostro snippet. Il suo scopo, abbiamo detto, sarà quello di generare un frammento di codice utile a trasformare una stringa in matrice di Byte da utilizzare come vettore o chiave di un algoritmo di criptazione. Nel nostro codice inseriremo anche alcune proprietà di sola lettura utili per restituire le matrici adatte per l’algoritmo Rijndael. Per prima cosa creeremo, con qualsiasi editor di testo, anche con lo stesso Visual Studio, un file XML con estensione .snippet. Se il file

UN TOOL PER LA CREAZIONE DI SNIPPET

All’indirizzo http://www.gotdotnet.com/Workspaces/Workspace.aspx?id=a927f4e7-8e7f-45ce-8b72-f3b9384a3eab è pubblicato uno strumento visuale che consente l’editing degli snippets. Personalmente lo ritengo ancora non sufficientemente maturo (a volte inspiegabilmente sparisce il codice inserito e ci sono anche altri Bug) e poi Visual Studio offre già il supporto Intellisense per l’editing degli snippets quindi perché complicare le cose?

verrà creato nella stessa directory che ospita gli altri snippet relativi alla sicurezza (nel mio caso "C:\Programmi\ Microsoft Visual Studio 8\VB\Snippets\1040\ Security") lo snippet apparirà accanto agli altri visti in precedenza. La sezione Header dello snippet apparirà quindi così:

```
<Header>
  <Title>Genera matrice di byte da Stringa</Title>
  <Author>Francesco Smelzo</Author>
  <Description>Trasforma una stringa in matrice di Byte da utilizzare come vettore o chiave di un algoritmo di criptazione e genera metodi di esempio</Description>
  <Shortcut>strByte</Shortcut>
</Header>
```

L'elemento Shortcut, lo ricordiamo, ci permette di definire una scelta breve da tastiera. Quando progettiamo uno snippet dobbiamo pensare all'utilizzo in un contesto. In questo caso il codice presuppone che ci sia, da qualche parte nella classe, una variabile o proprietà di tipo String da trasformare. A priori non possiamo saperne il nome quindi definiremo un segnaposto nell'elemento Snippet:

```
<Snippet>
   <Declarations>
     <Literal>
       <ID>strVarname</ID>
       <ToolTip>Nome variabile stringa</ToolTip>
       <Default>Password</Default>
     </Literal>
```

Tale elemento dovrà essere indicato, nel codice, tra i delimitatori $ e, nel frammento, assumerà inizialmente il valore indicato in Default. L'utente modificando tale valore e premendo il tasto Freccia Giù cambierà automaticamente il valore in tutte le parti del codice. Impostiamo, anche se non sarebbe strettamente necessario, altri due segnaposto: per il carattere di riempimento da utilizzare nel padding della stringa (se la password è più corta del numero di caratteri necessari alla matrice di byte la stringa viene riempita con tale carattere, altrimenti viene troncata) e per il tipo di encoding da utilizzare. Richiamiamo qui soltanto l'elemento Code, lo snippet completo (insieme ad altri che possono risultare utili) sono contenuti nel codice allegato:

```
<Code Language="VB" Kind="method decl">
    <![CDATA[
    Private Function $strVarname$ToBytes(maxLength as Integer) As Byte()
      If String.IsNullOrEmpty($strVarname$) Then
      $strVarname$="EmptyPassword"
      End If
      Dim b() as Byte
      b = $Encoding$.GetBytes($strVarname$.PadRight(maxLength, $PadChar$))
      Return b
    End Function

    Public ReadOnly Property RijndaelGetKey ()As Byte()
      Get
        Return $strVarname$ToBytes(32)
      End Get
    End Property
    Public ReadOnly Property RijndaelGetIV ()As Byte()
      Get
          Return $strVarname$ToBytes(16)
      End Get
    End Property
    ]]></Code>
```

Non resta, a questo punto che provare il nostro snippet. Nell'editor di Visual Studio, nel corpo di una classe digitiamo strByte + TAB e vediamo il nostro frammento apparire, come in Figura 8.

```
Private Function PasswordToBytes(ByVal maxLength As Integer) As Byte()
    If String.IsNullOrEmpty(Password) Then
        Password = "EmptyPassword"
    End If
    Dim b() As Byte
    b = System.Text.Encoding.UTF8.GetBytes(Password.PadRight(maxLength, "*"))
    Return b
End Function

Public ReadOnly Property RijndaelGetKey() As Byte()
    Get
        Return PasswordToBytes(32)
    End Get
End Property

Public ReadOnly Property RijndaelGetIV() As Byte()
    Get
        Return PasswordToBytes(16)
    End Get
End Property
```

Fig. 8: *Il nostro frammento inserito*

CONCLUSIONI

In questo articolo abbiamo visto la tecnologia degli snippets di Visual Studio. Uno strumento, tutto sommato, semplice e pratico da gestire che ci consente di mettere un po' d'ordine nel nostro codice e, possibilità non secondaria, condividerlo con altri.
Provare a sviluppare i vostri snippets e vedrete che non potrete più farne a meno!

Francesco Smelzo

L'AUTORE

Francesco Smelzo è specializzato nello sviluppo in ambiente Windows con particolare riferimento ad applicazioni in ambiente .NET sia web-oriented che desktop. Il suo sito web è www.smelzo.it. Come sempre è a disposizione per ricevere suggerimenti o richieste sull'articolo all'indirizzo di posta elettronica francesco@smelzo.it

APPLICAZIONI WEB CON GLI ACTIVEX

ECCO LE TECNICHE DA USARE PER SVILUPPARE SITI WEB DINAMICI UTILIZZANDO VISUAL BASIC. COME ESEMPIO APPLICATIVO DESCRIVEREMO COME PUBBLICARE SUL WEB UN CATALOGO DI FOTOGRAFIE USANDO UN ACTIVEX ED UN DATABASE ACCESS.

Conoscenze richieste
IIS, WebClass, Controlli Activex.

Software
Piattaforma Windows 2000 o superiore – IIS (Internet Information Server) - Visual Basic 6 SP6.

Impegno
Tempo di realizzazione
Supponiamo di volere sviluppare un sito web dinamico in Visual Basic. Come fare? Le soluzioni possibili sono molte. Potremmo scegliere di sviluppare un Isapi, overo un DLL che richiamata dal browser, con certi parametri produce in output una pagina particolare. Oppure potremmo usare DHTML, oppure una tecnologia quale quella degl Activex Document. In questo articolo utilizzeremo appunto quest'ultimo metodo per pubblicare un catalogo fotografico con tanto di browsing delle foto organizzato in categorie e sotto categorie.

CHE COSA È UN'ACTIVEX DOCUMENT

Sostanzialmente si tratta di una normale o quasi , applicazione molto simile ad un'applicazione tradizionale, il cui contenitore è però Internet Explorer. Per certi versi potete immaginarli come le vecchie applet Java. In realtà gli Activex Document sono delle applicazioni che possono essere eseguite in maniera generica attraverso dei contenitori di Activex. Ricordiamo che i contenitori principali di Activex sono Microsoft Internet Explorer e Microsoft Office Binder (raccoglitore di Office).
Gli Activex Document uniscono le funzionalità di un'applicazione alla flessibilità dei documenti. Gli Activex Document sono delle applicazioni che si comportano come dei documenti con contenuto dinamico. La compilazione di un progetto documento Activex produce il file che deve essere utilizzato nel contenitore di Activex e il file che fornisce le funzionalità (i servizi) al documento e quindi lo rende indipendente dal contenitore. La creazione di un Activex Document è analoga alla creazione di un Controllo Activex standard. Per i documenti Activex sono disponibili due tipi di progetto: Activex Document Dll (in-process component) e Activex Document Exe (out-of-process component). Il primo tipo di progetto ha come prodotto una DLL il secondo un file .EXE, in entrambi i casi però viene anche prodotto almeno un file con estensione .vbd che è quello che verrà mostrato nel contenitore. Il file .vbd all'atto della compilazione è posto nella stessa directory del file EXE o DLL prodotto. Naturalmente il file vbd senza il file EXE o DLL non può essere eseguito.

LE ALTERNATIVE AD ACTIVEX DOCUMENT

Le IIS e le DHTML application, sono una combinazione di codice Visual Basic e HTML, e sono eseguite in un Browser come le applicazioni HTML e ASP. Per implementare ed utilizzare un'applicazioni IIS è necessario un Server Web per le DHTML, invece, non è necessario dato che sono delle applicazione Client. Le IIS sono eseguite sul Server e rispondono alle richieste del Browser, le DHTML, invece, risiedono sullo stesso computer su cui si trova il Browser. Le IIS sono implementate con WebClass e pagine ASP (Active Server Pages). Come vedremo, le WebClass permettono d'implementare efficientemente e velocemente applicazioni Server per Internet.

CONTROLLI ACTIVEX

Come sappiamo, in Visual Basic, alla base della programmazione per componenti c'è l'architettura ad oggetti COM (Component Object Model) che fornisce un protocollo per definire dei componenti riutilizzabili. I controlli Activex definibili dall'utente rientrano in questa strategia. Con Visual Basic è possibile implementare diversi tipi di Activex tra i quali quelli pubblicabili su Internet che introdurremo nei successivi paragrafi. Alla base di un progetto Activex c'è l'oggetto UserControl esso, analogamente ad una form, presenta una finestra di progettazione, dove inserire i controlli che lo costituiscono e un modulo di codice, dove definire le proprietà, i metodi e gli eventi. Gli UserControl sono memorizzati in dei file con estensione ctl mentre dopo la creazione (Make) è costruito un file .OCX che può essere distribuito. Quando si crea un nuovo progetto Activex, le uniche proprietà disponibili sono quelle di base dell'UserControl, le altre proprietà possono essere aggiunte manualmente con la parola

chiave Property o attraverso il Wizard:“creazione guidata interfaccia controlli Activex” che si trova nelle aggiunte di Visual Basic.

IL CATALOGO FOTOGRAFICO

L’applicazione che permette di pubblicare un catalogo fotografico sarà costituita da un misto dato da un progetto Controllo Activex e un progetto IIS. Le foto verranno organizzate secondo una gerarchia con due livelli - categorie e sottocategorie - archiviata in un Database Access (nominato Catalogo). Per semplificare nel Database inseriremo una sola tabella nominata Catalogo descritta in Tabella 1.
Le immagini del catalogo, invece, non saranno salvate nel database ma in una directory locale o remota. Come vedremo avanti, la posizione di questa directory è specificata attraverso un parametro dell’UserControl. Le immagini sono associate ai record del database attraverso il loro nome che, per il nostro esempio, deve essere uguale alla denominazione della categoria. In questo modo ad ogni categoria è associata un’immagine se, invece, volessimo un’immagine per ogni sottocategoria le immagini dovrebbero essere nominate come categoria-sottocategoria, ecc. Nei paragrafi successivi introduciamo i progetti alla base dell’applicazione, questi esempi li trovate nel CD allegato alla rivista.

IL PROGETTO ACTIVEX

L’Activex per Internet della nostra applicazione, permette di scorrere e visualizzare delle immagini, il cui nome è specificato nella tabella Catalogo. Le posizioni del database e delle directory delle immagini verranno specificati con due parametri. Di seguito per punti descriviamo come creare il controllo Activex.

1 Create un nuovo progetto Activex (si controlli la figura 1) che referenzia la libreria MS ADODB 2.5

Fig. 1: **La struttura di un progetto Activex**

2 Sull’UserControl inserite i controlli che permettono di visualizzare e scorrere i dati contenuti nel database. In particolare inserite: 3 textbox (nominati TxtCategoria, TxtSottoCategoria e TxtNote) per visualizzare il contenuto dei campi Categoria, Sottocategoria e Note; una PictureBox

Catalogo	
Campo	**Tipo**
Numero	**Contatore**
Categoria	**Testo (50)**
SottoCategoria	**Testo (50)**
Note	**Testo (250)**

Tabella 1: La tabella catalogo

(PictureAnteprima) per visualizzare l’anteprima dell’immagine collegata al record; 2 CommandButton (avanti e dietro) e un TextBox (TxtNumeroElementi) per controllare la sequenza dei record visualizzati. Tra l’atro prevedete una Label (LabelIE) da utilizzare per visualizzare, e cliccare, il path dell’immagine in anteprima. In particolare, cliccando su questa label faremo in modo che l’immagine si apra in Internet Explorer. Disponete il tutto come in figura.

Fig. 2: **L’UserControl del progetto Activex**

3 Nella parte dichiarativa dell’UserControl, bisogna definire le seguenti variabili:

```
'La parte dichiarativa...
Private I As Integer
Private Rstcatalogo As ADODB.Recordset
Private pathimmaginehtml As String
Private mvarpathdatabase As Variant
Private mvarpathdirimmagini As Variant
```

La variabile I indica la posizione relativa del record corrente; Rstcatalogo è il RecordSet caricato in memoria: pathimmaginehtml è il Path dell’immagine in anteprima; pathdatabase e pathdirimmagi-

ni sono i parametri che il controllo riceverà dalla pagina HTML che lo contiene. In particolare pathdirimmagini rappresenta il path della directory che contiene le immagini del catalogo, pathdatabase, invece, è il path del database Catalogo.

4 Il codice per la definizione delle Property Get e Let dei parametri dell'UserControl è il seguente.

```
Public Property Get pathdirimmagini() As Variant
    pathdirimmagini = mvarpathdirimmagini
End Property
Public Property Let pathdirimmagini(ByVal
    vNewValue As Variant)
    mvarpathdirimmagini = vNewValue
End Property

Public Property Get pathdatabase() As Variant
 pathdatabase = mvarpathdatabase
End Property
Public Property Let pathdatabase(ByVal
    vNewValue As Variant)
    mvarpathdatabase = vNewValue
End Property
```

CARICAMENTO DI UN 'IMMAGINE IN MODALITÀ ASINCRONA

Il caricamento di un'immagine in un UserControl può essere fatto anche in modo asincrono (cioè caricare l'immagine e nel frattempo effettuare altre operazioni). Questa caratteristica viene regolata dalle proprietà AsyncReadComplete e AsyncReadProgress. La sintassi del metodo AsyncRead è la seguente: object.AsyncRead Target, AsyncType [, PropertyName], [AsyncReadOptions] Target è l'URL da cui si esegue il DownLoad delle proprietà, nei nostri esempi dovrebbe essere il path dove si trova l'immagine da caricare. AsyncType è il tipo di proprietà di cui bisogna fare il DownLoad, i possibili valori sono: vbAsyncTypePicture (nel caso di un'immagine), vbAsyncTypeFile, vbAsyncTypeByteArray. PropertyName è il nome delle proprietà di cui si esegue il Download.

5 Per gestire i valori dei parametri utilizziamo gli eventi UserControl_WriteProperties e UserControl_ReadProperties, nei quali prevediamo il seguente codice.

```
Private Sub
    UserControl_ReadProperties(PropBag As
    PropertyBag)
 pathdatabase =
    PropBag.ReadProperty("pathdatabase")
 pathdirimmagini =
    PropBag.ReadProperty("pathdirimmagini")
End Sub
Private Sub
    UserControl_WriteProperties(PropBag As
    PropertyBag)
 Call PropBag.WriteProperty("pathdatabase",
    App.Path)
 Call PropBag.WriteProperty("pathdirimmagini",
    App.Path + "/images")
End Sub
```

Con UserControl_ReadProperties si leggono i valori delle Property forniti dal contenitore dell'UserControl, mentre con WriteProperties s'impostano i valori di Default.

6 Di seguito presentiamo il codice per collegarsi al database e per eseguire la query che legge tutte le categorie in catalogo. Questa query è impostata nell'evento UserControl_Show che è generato quando è visualizzato il controllo.

```
Private Sub UserControl_Show()
 If pathdirimmagini = "" Then
  pathdirimmagini = App.Path
 End If
 If pathdatabase = "" Then
  pathdatabase = App.Path
 End If
 initdatabase ("select * from catalogo " _
 & "order by categoria")
End Sub
Private Sub initdatabase(str As String)
 Dim strcnn As String
 strcnn = "Provider=microsoft.jet.oledb.4.0;" _
 & "Data Source=" + " " & pathdatabase _
 & "\catalogo.MDB; "
 Set Rstcatalogo = New ADODB.Recordset
 Rstcatalogo.CursorType = adOpenKeyset
 Rstcatalogo.LockType = adLockOptimistic
 Rstcatalogo.Open str, strcnn, , , adCmdText
 impostadati (1)
End Sub
```

Notate che alla initdatabase la query è passata con la stringa Str. Il codice della initdatabase si collega al database ed in base alla query carica Rstcatalogo, infine invoca la impostadati per caricare i dati nei controlli dell'UserControl. La impostadati è descritta di seguito.

7 Il codice dell'impostadati e quello per scorrere i Record con i pulsanti avanti e dietro è il seguente.

```
Private Sub impostadati(Optional ind As Variant)
 If Not IsMissing(ind) Then
  I = ind
 End If
 If Not Rstcatalogo.EOF Then
  With Rstcatalogo
   txtcategoria = !categoria
   txtsottocategoria = !sottocategoria
   txtnota = !note
```

```
    LabelIE.Visible = False
    DocPicture.Picture = Nothing
    txtnumeroelementi = CStr(I) + "/" +
                                            CStr(.RecordCount)
    On Error GoTo fine
    DocPicture.Picture = LoadPicture(pathdirimmagini
                                                          & _
    "\" & txtcategoria + ".jpg")
    pathimmaginehtml = pathdirimmagini & _
    "\" & txtcategoria + ".jpg"
    LabelIE.Caption = ">> " + pathimmaginehtml
    LabelIE.Visible = True
fine:
  End With
 End If
End Sub
Private Sub Avanti_Click()
 If Not Rstcatalogo.EOF Then
  On Error GoTo fine
  I = I + 1
  If I > Rstcatalogo.RecordCount Then
   I = Rstcatalogo.RecordCount
  Else
   Rstcatalogo.MoveNext
   impostadati
  End If
 End If
fine:
End Sub
Private Sub Dietro_Click()
 If Not Rstcatalogo.EOF Then
  I = I - 1
   If I <= 0 Then
    I = 1
   Else
    On Error GoTo fine
    Rstcatalogo.MovePrevious
    impostadati
   End If
 End If
fine:
End Sub
```

La impostadati inserisce i valori nei TextBox, nel PictureBox ed imposta la LabelIE. In particolare txtnumeroelementi indica il record corrente sul numero dei record caricati in Rstcatalogo. Il valore di txtnumeroelementi sarà modificato in base al valore della variabile I.

8 Per ricercare un particolare Record e per visualizzare l'immagine con Internet Explorer, utilizziamo il seguente codice.

```
Private Sub Cerca_Click()
 Dim QUERY As String
 QUERY = "Select * from catalogo where " _
 & "categoria " & "like '%" & txtcategoria & "%'"
 If txtsottocategoria <> "" Then
  QUERY = QUERY & " and sottocategoria='" _
  & txtsottocategoria & "'"
 End If
 CancellaCampi
 initdatabase (QUERY)
End Sub
Private Sub CancellaCampi()
 txtcategoria = ""
 txtsottocategoria = ""
 txtnota = ""
 LabelIE.Visible = False
 DocPicture.Picture = Nothing
End Sub
Private Sub LabelIE_Click()
 Dim IEApp As Object
 Set IEApp =
            CreateObject("InternetExplorer.Application")
 IEApp.Visible = True
 IEApp.Navigate pathimmaginehtml
 IEApp.Height = 500
 IEApp.Width = 500
 IEApp.MenuBar = False
 IEApp.toolBar = False
End Sub
```

TAG SCRIPT

In una pagina HTML è possibile inserire del codice VBScript utilizzando il Tag Script. Per esempio, se create un file HTML ed in esso inserite le seguenti istruzioni, potete aprire il sito www.ioprogrammo.it.

```
<HTML>
 <script language="Vbscript">
  Dim max
  max= msgbox ("Salve, vuoi aprire il sito" _
  & "di IoProgrammo",vbokcancel,"Esempio")
  if max=vbok then
   open "http://www.ioprogrammo.it"
  else
   Msgbox "ciao"
  end if
 </script>
</HTML>
```

PAGINA HTML PER USERCONTROL

Per utilizzare un UserControl in una pagina HTML, possiamo procedere nel seguente modo.

1 Creare l'OCX con il commando File/Crea Progetto1.OCX (facciamo notare, però, che il progetto Activex nel nostro esempio è stato

nominato Catalogo).

2 Create una pagina HTML, simile alla seguente, con i riferimenti al controllo Progetto1.OCX.

```
<HTML>
 <BODY>

 <OBJECT
  classid=clsid:44BCAF42 ...>
   <param name="pathdatabase" value="C:\data">
   <param name="pathdirimmagini"
                          value="C:\data\images">
 </OBJECT>
 </BODY>
</HTML>
```

INTERNET COMPONENT DOWNLOAD

Internet Component Download (ICD) è il meccanismo che regola il downloading e l'istallazione del codice per internet. ICD è usato da Internet Explorer per il Download automatico e l'istallazione controllata del codice presente su Internet. In parole povere ICD si preoccupa di associare al controllo, presente sulla pagina, le DLL di supporto. Per esempio se su una pagina HTML è pubblicato un controllo Activex, che fa uso di alcuni controlli di base di Visual Basic, per la corretta esecuzione nel Browser sono necessari i file di RunTime di Visual Basic (Msstkprp.dll e VB6 Runtime and OLE Automation). Di come distribuire questi file, però, il programmatore non si deve preoccupare dato che ci pensa il Wizard di creazione guidata del pacchetti di installazione.

3 Per creare la pagina HTML precedente, potete utilizzare un editor come Blocco Note salvando il file ottenuto con estensione htm. Notate che per inserire il controllo OCX in una pagina HTML bisogna utilizzare i Tag Object e Param. Classid, invece, è l'identificatore univoco del controllo. Naturalmente il clsid deve essere quello associato al vostro

Fig. 3: **La finestra per impostare il Debug del controllo**

UserControl. Un modo banale per ricavare il clsid è copiarlo dalla pagina HTML creata da Visual Basic nella fase di Debug del controllo (selezionando sul Browser mostra origine HTML), oppure seguire le indicazioni fornite dal Wizard d'installazione. Infine notate che con i tag PARAM si specificano i valori dei parametri passati dalla pagina HTML al controllo.

IL PROGETTO IIS

Un controllo OCX per Internet può essere utilizzato su una semplice pagina HTML o ASP, come visto in precedenza, oppure inserito in una WebClass di un progetto IIS. Per passi illustriamo come creare un progetto IIS che utilizza il nostro OCX.

1 Create un Progetto IIS e salvatelo in una directory dedicata.

2 Create una pagina HTML simile a quella presentata nei paragrafi precedenti e salvatela nella stessa directory in cui avete salvato il progetto. Nel nostro caso la pagina è stata nominata DocumentiWeb.htm e la trovate nel CD allegato alla rivista.

3 Sulla maschera di progettazione della WebClass, selezionate il bottone (oppure utilizzate il tasto destro Mouse sul contenitore oggetti WebItem) aggiungi WebItem HTML, nella finestra che compare, selezionate il file HTML che avete creato in precedenza.

Fig. 4: **La finestra per impostare il modello HTML**

Dopo i passi precedenti sulla parte destra della finestra di progettazione della WebClass, compaiono dei riferimenti agli elementi principali della vostra pagina HTML. Questi elementi possono essere utilizzati a livello di programmazione solo dopo averli connessi a qualche evento personalizzato o a qualche WebItem.

4 Le prime istruzione da eseguire all'avvio del progetto IIS, devono essere inseriti nell'evento WebClass_Start.

```
Private Sub WebClass_Start()
 Set NextItem = Template1
End Sub
```

Start insieme a Initialize, Begin Request, End Request, Terminate sono gli eventi fondamentali della WebClass. La proprietà NextItem serve ad indicare qual'è il successivo WebItem da eseguire, in altre parole permette il passaggio dell'elaborazione tra WebItem. Nel nostro caso il controllo è passato al WebItem Template1, cioè alla pagina HTML che abbiamo creato nel punto 2.

5 Dopo che il controllo è passato al Template1 dobbiamo stabilire le azioni da eseguire. A tale proposito possiamo utilizzare il primo evento del WebItem eseguito, in pratica Respond che è l'evento predefinito degli oggetti WebItem.

```
Private Sub Template1_Respond()
 With Response
  .Write "<html>"
  .Write "<body>"
  .Write "<a href =http:\\www.ioprogrammo.it> " _
  & "<font color=#EF8B1C> HOME
                                    ioProgrammo</a>"
  .Write "</body>"
  .Write "</html>"
 End With
 Template1.WriteTemplate
End Sub
```

Nell'evento Template1_Respond, tra l'altro, è utilizzato il metodo WriteTemplate che permette d'inviare il modello HTML al Browser (dopo averlo elaborato). A questo punto il controllo OCX è mostrato sul Browser. Notate che sul Browser, oltre al controllo è inserito un collegamento al sito www.ioprogrammo.it. Naturalmente per rendere funzionante l'esempio dovete impostare opportunamente i valori dei parametri del UserControl.

Fig. 5: Il Progetto IIS in fase di esecuzione

WEB SERVER

Per poter creare o eseguire delle applicazioni IIS è necessario installare il Server Web compatibile con il sistema operativo della vostra macchina. Gli esempi presentati in questo articolo, sono stati sviluppati con la piattaforma Windows XP Professional – Internet Information Server (IIS) 5.1. Questi, però, con alcuni accorgimenti possono essere utilizzati anche con altri sistemi operativi. Ricordiamo che nel caso di Windows 95/98 il Server Web è Personal Web Server (PWS) che si trova nella suite Visual Studio oppure nella cartella add-ons del Cd del sistema operativo. Attenzione al fatto che il PWS installato deve essere nella stessa lingua di Visual Basic!

CONCLUSIONI

Nell'articolo non abbiamo dato spazio alle tecniche di protezione e sicurezza, tuttavia è innegabile che temi come questi siano da tenere in dovuto conto quando si sviluppano applicazioni per la rete. Prima di "deployare" un nuovo progetto, vi consigliamo sempre di eseguire uno studio approfondito dei possibili problemi legati ad applicazioni che devono essere esposte ad un largo numero di utenti.

Massimo Autiero

IIS E DIRECTORY VIRTUALI

Fig. 1: La finestra nuovo progetto Visual Basic

Le applicazioni IIS, o progetti IIS, si creano dalla finestra NuovoProgetto scegliendo Applicazione IIS, il progetto che si ottiene è di tipo DLL Activex e ha come componente di avvio un oggetto chiamato WebClass1. All'avvio del progetto IIS nel Web Server viene creata una directory virtuale il cui nome può essere scelto dal programmatore. Questa directory nei Browser è visualizzata come sottodirectory della directory principale del Server Web. Se il progetto Visual Basic non è salvato, la directory Virtuale assegnata dall'ambiente d'implementazione è la Temp. Dopo la creazione è possibile modificare il nome della directory virtuale ma non la sua posizione. Per vedere la directory e le sue caratteristiche (Alias, permessi ...) basta selezionare le proprietà del Web Server e cliccare sul bottone impostazioni avanzate. S'intuisce che un'applicazione IIS non può funzionare senza Web Server.

Fig. 2: La directory Virtuale del progetto IIS della nostra applicazione

PERSISTENZA DEI DATI CON CAYENNE

NUMEROSI FRAMEWORK ORM PERMETTONO DI CREARE OGGETTI A PARTIRE DA BASI DI DATI E VICEVERSA. CAYENNE HA DELLE CARATTERISTICHE ESTREMAMENTE INTERESSANTI E UTILIZZABILI SIA PER PROGETTI AMATORIALI CHE PROFESSIONALI

Qual è il problema dei database moderni? Sostanzialmente nessuno. Tutti sono affidabili, veloci e ciascuno espone caratteristiche degne della tecnologia moderna. Ciò che unisce tutti i database è il fatto che sono pensati secondo un modello relazionale. I dati all'interno di un database sono organizzati in tabelle, ciascuna tabella è costituita da righe. Ciascuna riga è costituita da colonne. Le tabelle sono messe in correlazione fra loro tramite un accoppiamento di colonne determinate.
Potete pensare a un database come a un file excel. I fogli che compongono il progetto Excel potrebbero essere assimilati a tabelle, le righe del foglio a record, le intestazioni a colonne.
Ora, il punto è che i dati all'interno di una database vengono manipolati tramite un linguaggio "SQL" che ragiona in termini di relazioni fra elementi delle tabelle, ad esempio:

```
Select cmn.* , abt.* from comuni as cmn,
abitanti as abt where
abt.ID_COMUNE=cmn.ID_COMUNE;
```

Un'espressione del genere per essere utilizzata all'interno di un linguaggio ad alto livello, dovrebbe essere utilizzata da un metodo "query" che esegue l'istruzione e ritorna una struttura idonea ad essere gestita dal linguaggio. Questo tipo di approccio ha un enorme svantaggio, ovvero le colonne del db, così come le tabelle, così come le righe non sono oggetti dotati di proprietà e metodi bensì sono semplicemente un "dato" acquisito e manovrabile solo ancora una volta tramite istruzioni SQL. Per aggirare questo problema nascono gli ORM ovvero framework intermedi che si occupano di "mappare" le tabelle, le righe e le colonne, in corrispondenti oggetti manipolabili direttamente dal linguaggio.
In questo articolo ci occuperemo di Cayenne, un ORM piuttosto avanzato che offre funzionalità di tutto rilievo.

UN CASO CONCRETO: GESTIAMO UN BLOG

Si supponga di voler realizzare un'applicazione Web per la gestione/visualizzazione di un blog. La nostra applicazione farà quello che fanno più o meno tutti i blog, sarà possibile inserire nuovi post (ognuno è caratterizzato da una data, un flag che indica la presenza o meno in home page, un titolo, un abstract e un contenuto vero e proprio). Per ogni post sarà possibile inserire dei commenti (specificando l'autore; la data del commento è quella del sistema).
Prevederemo una pagina introduttiva, una di autenticazione, una di risorse ed una di visualizzazione dei post del blog. Nella pagina di visualizzazione del blog, dapprima saranno visibili solo alcuni post (quelli con flag "in_home" a "true") di cui verranno mostrati solo titolo e abstract. Ognuno di essi avrà un link che condurrà a una pagina dove sarà mostrato l'interno post. Dalla pagina del blog dovrà essere possibile tramite un link, la visualizzazione dei titoli/abstract di tutti i post inseriti. L'amministratore potrà eliminare sia dei post sia singoli commenti. Inoltre un utente autenticato come amministratore a mezzo di una form di login potrà inserire nuovi post e modificare quelli esistenti

PRIMA LA BASE DATI O IL MODELLO AD OGGETTI?

Cayenne dispone di un tool grafico dal quale è possibile sia modellare le classi e conseguentemente di generare script per il database, sia generare le classi a partire da una base dati esistente. In questo articolo seguiremo il secondo approccio. Questo perché di solito chi crea la base dati è un esperto di database ma non necessariamente un esperto di Cayenne. Pertanto è fondamentale creare una "buona"

base dati con gli strumenti con cui si ha più dimestichezza.
Nella cartella dove è stato scompattato Cayenne è presente una cartella bin/. Al suo interno ci sono degli script per lanciare il modeler (.bat per sistemi Windows, .sh per quelli Linux). Esso si presenta con una console grafica.
Prima di eseguire la creazione delle classi è bene inserire le informazioni per la connessione al database selezionando Tools > Preferences. Su "Classpath" impostimo il file jar dei driver del database che si vuole utilizzare; su "Local DataSources" impostiamo le stringhe di connessione JDBC al database. Per verificare che tutto funzioni utilizziamo il pulsante "Test"; se appare la scritta "Connected Succesfully" si è impostato tutto correttamente. Per rendere persistenti queste impostazioni è sufficiente usare il pulsante "Save".
Ora si può creare un nuovo progetto; da "File" scegliendo "New Project" oppure aprire un progetto esistente. Per quanto riguarda questo articolo utilizzeremo un progetto esistente, disponibile nel codice allegato al CDRom, il file da aprire è cayenne.xml. Il primo nodo della gerarchia ("CayenneIoProgrammo") è il nome del progetto. A seguire il DataNode creato e il DataMap associato. Il DataMap è il "cuore" della configurazione, ove risiedono tutte le informazioni di associazione tra tabelle e oggetti (entrambi gli oggetti sono inclusi nel datamap).
Per creare altri data node è sufficiente fare clic con il mouse sul nome del progetto e da "Project" scegliere "Create DataNode". Lasciare tutti i valori di default a meno della voce "Local dataSource": selezionare quanto definito in precedenza sulle preferenze del tool. Fatto questo premere sul pulsante "Sync with local": i parametri di connessione JDBC vengono reperiti da quanto precedentemente impostato.
Al termine di ogni modifica, per renderle persistenti, salvare il progetto con "File > Save" e selezionare il percorso dove memorizzarlo.
Sul CD-Rom è presente anche uno script per la creazione della base dati per MySql.

UN NUOVO PROGETTO IN ECLIPSE

Le classi generate possono essere utilizzate e compilate sia usando i tool standard che utilizzando il proprio IDE preferito. Usando Eclipse si può creare un nuovo progetto; avendo installato il plug-in WTP si può creare un Dynamic Web Project specifico per il Tomcat installato. Non resta che copiare le classi generate sotto la cartella src/ del progetto e i file XML generati (in particolare i file cayenne.xml, BlogMap.map.xml e CayenneIoProgrammo Node.driver.xml) sotto la cartella /WebContent/WEB-INF/classes/. Infatti, affinché un'applicazione possa utilizzare Cayenne, i file di configurazione devono stare sotto una cartella che è o nel classpath o in un altro posto accessibile (al run-time) alle classi. Infine copiare il file cayenne.jar (si trova nella cartella lib/ dell'installazione del pacchetto) nella cartella /WebContent. Eseguendo un refresh su Eclipse il progetto includerà le nuove classi e, se tutto è stato configurato correttamente, le compilerà.

Fig. 1: Il progetto di esempio aperto con Eclipse 3.1.

UTILIZZARE CAYENNE: DATACONTEXT

L'oggetto fondamentale da conoscere, usando Cayenne, è il DataContext. Attraverso il DataContext si possono eseguire le interazioni con il database (leggere i dati, eliminare record, eseguire query e così via). Il DataContext deve leggere i file xml generati precedentemente dal tool grafico. Per farlo è necessario che i file siano accessibili dal classpath. Considerando che una webapp ha settato, di default, il classpath sulla cartella WEB-INF/classes, tale cartella è una buona candidata per copiare i file di configurazione.
Una volta resi accessibili i file di configurazione è necessario inizializzare il DataContext facendoglieli leggere. Una Webapp dovrebbe prevedere sempre una servlet di inizializzazione (meglio se questa viene eseguita automaticamente allo start-up di tomcat) che la inizializza in questo modo:

```
public class initServlet extends HttpServlet {
```

```
  //...
    public void init(ServletConfig conf) throws
                                    ServletException {
        super.init(conf);
BasicServletConfiguration.initializeConfiguration(
          conf.getServletContext() );
        //...
    }
  //...
}
```

Nel progetto la servlet che effettua l'inizializzazione è it.ioprogrammo.cayenne.servlet. InizializzaCayenne.java. Per come la abbiamo strutturata è leggermente più complessa rispetto alla sola inizializzazione del DataContext. Infatti va a leggere un file di properties esterno da cui recupera il nome del file xml da caricare. Questo è utile in tutti quei contesti, comuni in progetti deployati su più sistemi, in cui diverse installazioni accedono a diverse basi di dati.
Una volta inizializzato il DataContext (fatto una volta per tutte!) lo si può recuperare in una qualsiasi pagina jsp utilizzando la sessione (oggetto predefinito "session"):

```
DataContext context =
 BasicServletConfiguration.getDefaultContext(
  session
 );
```

LA PRIMA QUERY

Lavorare con dati dinamici significa, essenzialmente, reperirli da una base dati; l'operazione che permette di eseguire una query è:

```
List risultato = context.performQuery( .. );
```

DOWNLOAD E INSTALLAZIONE DI CAYENNE

Collegandosi alla home page del framework, http://objectstyle.org/cayenne, si può seguire il link "Download" e quindi scegliere la versione (al momento della stesura dell'articolo la versione stabile è la 1.1.4). Scelto uno dei mirror viene eseguito il download del framework: esso si presenta in formato .tar.gz (TAR compresso); per scompattarlo si deve utilizzare o il comando Linux:

```
tar -xzf nomefile.tar.gz
```

Oppure, sotto Windows, si può utilizzare un tool come WinRar o WinZip. La cartella risultate ha la struttura mostrata in FIGURA.

Tra i parametri ci può essere una qualunque oggetto derivato da GenericServletQuery. Uno degli oggetti senz'altro più utile è una query su una tabella; per indicare la tabella si può usare la classe; ecco, per esempio, come creare una query sulla tabella blog_entry:

```
SelectQuery mySelectQuery = new
                    SelectQuery(BlogEntry.class);
```

Spesso è necessario indicare dei valori per uno o più campi. In questi casi la query può essere costruita come:

```
new SelectQuery(BlogEntry.class, espressione);
```

dove espressione è un oggetto di tipo Expression; esso può essere costruito a partire da una stringa:

```
Expression esp = Expression.fromString(
                              SELECT_STRING);
```

dove SELECT_STRING può essere una qualunque stringa del tipo "campo=valore" (per date e numeri) o "campo like 'valore' " per stringhe. Un modo per nominare i campi è quello di ricorrere ai nomi delle property generati in automatico da Cayenne; per esempio:

```
SELECT_STRING =
        BlogEntry.IN_HOME_PROPERTY+" like 'una
                                     stringa'";
```

Quando si vuole costruire stringhe di selezione con parametri variabili, si usa la notazione “$valore” per indicare dei segnaposto da sostituire con valori opportuni; per esempio:

```
SELECT_STRING =
BlogEntry.IN_HOME_PROPERTY+ " like $var"+
              BlogEntry.DATA_PROPERTY+"=$data";
```

per "istanziare" il segnaposto si può costruire un oggetto Map; su tale oggetto si assegnano coppie chiave/valore; i nomi della chiave vanno a sostituire i segnaposto omonimi; per esempio:

```
Map p = new Map();
p.put("var", "Sì");
p.put("data", "2006/01/01");
esp = esp.expWithParameters( p );
```

equivale a scrivere:

```
SELECT_STRING =
  BlogEntry.IN_HOME_PROPERTY+
```

```
  " like 'Sì'"+
  BlogEntry.DATA_PROPERTY+
  "=2006/01/01";
```

Che accadese se, per esempio, non si assegnasse alcun valore a "data"? Semplicemente quella parte di espressione verrebbe ignorata! Si può costruire una espressione che contiene valori per ogni campo, dove sia i nomi dei campi sia quelli dei valori sono ricavati dai nomi delle proprietà:

```
static final String SELECT_STRING =
 BlogEntry.IN_HOME_PROPERTY+" like $"
 +BlogEntry.IN_HOME_PROPERTY+
 " and db:id = $id and titolo like $titolo and " +
 BlogEntry.DATA_PROPERTY+"=$"+
 BlogEntry.DATA_PROPERTY+" and "+
 BlogEntry.ABSTRACT_PROPERTY+
" like $"+BlogEntry.ABSTRACT_PROPERTY;
```

Tale stringa può essere costruita una volta per tutte e dichiarata statica all'interno delle classi affinché non venga costruita ogni volta ("sprecando" risorse nella costruzione di nuovi oggetti stringa).
Un modo molto comodo per sostituire tutti i segnaposto è, per le applicazioni Web, ricavare un oggetto Map direttamente dai parametri della request e usare il metodo expWithParameters per sostituire tutti i segnaposto di una espressione:

```
Map params = request.getParameterMap();
esp = esp.expWithParameters( params );
```

Usando applicazioni Web, particolare attenzione va fatta, come sempre, per i tipi di dati che non sono stringhe: Integer (per essi basterà eseguire il parsing corretto) e le date (in questo caso va fatta attenzione alla possibilità che l'applicazione Web possa accettare date in diversi formati; per esempio GG/MM/AAAA per utenti italiani e MM/GG/AAAA per quelli americani e così via...). Per evitare che gli utenti siano costretti a scrivere le date in formato JDBC, nell'applicazione di esempio si è fatto uso di un "remapping" di tutte le date creando un'apposita funzione:

```
Map params = Common.remap(
 request.getParameterMap());
```

Su una query si può sia impostare un ordinamento sui risultati (ascendente o descendente) sia limitare il numero di record restituiti che impostare una paginazione:

```
qry.addOrdering(
 BlogEntry.DATA_PROPERTY,
  ascending);
qry.setFetchLimit(numRecord);
qry.setPageSize(dimPagina);
```

Infine si può assegnare un nome ad una query e far sì che il risultato della sua esecuzione sia mantenuto in cache: ogni sua "ri-esecuzione" non comporta una nuova query sul database, ma il reperimento dei dati dalla cache, a tutto vantaggio della performance!

INSERIRE NUOVI RECORD

Per creare nuovi record della base dati si ricorre, ancora una volta, al DataContext di cui si invoca il metodo createAndRegisterNewObject sulla classe dell'oggetto da creare:

```
BlogEntry ris = (BlogEntry)
  context.createAndRegisterNewObject(
  BlogEntry.class );
```

poi si assegnano i valori alle proprietà e si esegue l'inserimento vero e proprio invocando il metodo:

```
  context.commitChanges();
```

ELIMINARE/AGGIORNARE RECORD ESISTENTI

Per eliminare un record è sufficiente eseguire il metodo deleteObject (sempre di DataContext) a cui si passa l'oggetto da eliminare. Per aggiornare un oggetto esso va reperito e vanno riassegnate le sue proprietà; nell'articolo è stato creato un metodo (getOne) usato per reperire un BlogEntry dal suo codice:

```
Object id = request.getParameter("id");
BlogEntry record = getOne(context, id);
if (record!=null) {
 setFromRequest(record, request);
 context.commitChanges();
}
```

REPERIRE LA CHIAVE DI UN RECORD

Osservando il codice sorgente precedente può sorgere un dubbio: come si fa a reperire l'identificativo di chiave del record, se esso non appare tra le proprietà esposte dalla classe? In questo caso viene in aiuto un metodo della classe DataObjectUtils e precisamente il metodo stati-

co intPKForObject :

```
int chiave =
 DataObjectUtils.intPKForObject( blog );
```

GESTIRE RECORD "DIPENDENTI"

Nel caso si vogliano aggiungere dei commenti ad un blog esistente, per prima cosa si deve reperire l'oggetto associato al record (usando, per esempio, una ricerca per chiave) e successivamente creare e aggiungere un oggetto di tipo Comment; per aggiungerlo si utilizzi il metodo AddToCommentArray:

```
recordBlog.addToCommentArray( commento );
```

Invece, per reperire tutti i commenti associati ad uno specifico blog:

```
Comment[] commenti = blog.getCommentArray();
```

ESEGUIRE DEL CODICE SQL...

Cayenne, a differenza di altri framework, lascia comunque libero il programmatore di eseguire comandi SQL di qualsiasi tipo. Infatti è sempre possibile usare oggetti di tipo SQLTemplate:

```
SQLTemplate templ = new SQLTemplate(
(DataMap)dc.getDataMaps().iterator().next(),
            "SELECT descr_ita, descr FROM News,
                         Autori WHERE Autori.id=news.id",
            true);
templ.setFetchingDataRows(true);
List ris=dc.performQuery(templ);
```

Gli oggetti restituiti da un simile comando (avendo impostato la proprietà FetchingDataRows a true) sono di tipo org.objectstyle.cayenne.DataRow.

ALCUNI SUGGERIMENTI

Con qualche accorgimento possiamo migliorare la qualità del codice sviluppato. Per prima cosa evitiamo di scrivere i nomi dei campi come stringhe (per esempio come nomi di campi di una form che poi vengono letti e usati nelle select); questo per evitare di disseminare errori del tipo "desrizione" anziché "descrizione": molto meglio usare le costanti definite negli oggetti creati da Cayenne (per esempio BlogEntry.DESCRIZIONE_PROPERTY). Questa accortezza agevola anche l'individuazione dei punti da modificare quando viene eliminato un campo da una tabella e viene rigenerato l'oggetto che esegue il mapping. Nelle form è sempre meglio inserire l'indicazione del numero massimo di caratteri accettati da un campi di input. Anziché scrivere i valori usare delle costanti; per esempio: BlogEntry.DESCRIZIONE_MAXCHARS. Queste vanno inserite negli oggetti creati da Cayenne (ma non in quelli il cui nome inizia con "_": questi vanno lasciati sempre inalterati!).
Infine consiglio di creare sempre delle classi "manager" per la gestione delle usuali operazioni (select, insert, delete, ...) e che si occupano del mapping tra i parametri passati sulla request e le proprietà degli oggetti creati (o eliminati, reperiti e così via).
Analizzando la webapp allegata all'articolo si noterà l'uso di tutti questi accorgimenti.

CONCLUSIONI

In Figura l'applicazione Web risultante.

Fig. 2: *L'applicazione Web risultante*

Cayenne è un prodotto stabile, dalle ottime caratteristiche sia per ambienti di produzione che necessitano di prestazioni elevate sia per progetti semplici dove la facilità d'uso degli strumenti è una delle caratteristiche preponderanti.
Da poco (marzo 2006) Cayenne è stato inserito tra i progetti di Apache; seguendo la solita trafila definita dall'associazione, inizialmente è considerato un progetto allo stato di incubazione (incubator). Dopo un periodo di prova, in cui dimostrerà se è capace di creare una nutrita comunità attorno al prodotto, in grado di garantire stabilità allo sviluppo del progetto stesso, diventerà a tutti gli effetti uno dei progetti di Apache. Un motivo d'orgoglio per l'iniziatore del progetto e un'ulteriore garanzia di stabilità e di una solida comunità per tutti gli utenti...

Ivan Venuti

SCRIVILO SUL MURO CON SPRING MVC

IMPARIAMO COME COSTRUIRE APPLICAZIONI PERFETTAMENTE MANUTENIBILI E CON UN ALTO GRADO DI DISACCOPPIAMENTO FRA CODICE E LAYOUT. REALIZZIAMO IL TUTTO CON UN ESEMPIO UTILE: UN GUESTBOOK CHE "SCRIVE SU UN MURO"...

Come abbiamo avuto già modo di apprendere nel numero 100 di ioProgrammo, Spring è un framework che viene utilizzato sia per facilitare l'utilizzo di altri framework sia per scrivere codice più pulito e manutenibile. Quest'ultima caratteristica è determinata dall'adozione di un pattern chiamato Inversion of Control che consente di comporre i vari moduli di un progetto tramite un file XML. Per meglio comprendere questo approccio alla programmazione analizziamo brevemente il seguente codice:

```
public interface IA{
   public void actionA();
}
public interface IB{
   public void actionB();
}

public class B implements IB{
   public void actionB(){
      System.out.println("action B");
   }
}

public class A implements IA{
protected IB b;
   public void setB(IB b){
        this.b=b;
   }

   public void actionA(){
      System.out.println("action A");
       this.b.actionB();
  }

  public static void main(String [] ar){
     ClassPathXmlApplicationContext f=new
     ClassPathXmlApplicationContext("factory.xml");
        IA a=(IA)f.getBean("aa");
      a.actionA();
  }
}
```

```
// file factory.xml
<?xml version="1.0" encoding="UTF-8"?>
<beans>
<bean id="bb" class="B"/>
<bean id="aa" class="A">
    <property name="b" >
       <ref bean="bb" />
     </property>
</bean>
</beans>
```

Abbiamo definito due interfacce e due classi che le implementano. La particolarità risiede però nell'implementazione del main presente nella classe A e nell'utilizzo del file factory.xml che determina i collegamenti tra le classi utilizzate. Infatti nel main viene definito un ApplicationContext dal quale è possibile ottenere un'istanza dell'interfaccia IA, l'oggetto f definisce una factory di oggetti creati e composti secondo i dettami definiti nel file XML. In questo file è definito un tag <beans> che racchiude diversi tag <bean>; ogni tag <bean> definisce un oggetto istanza della classe definita dall'attributo class. L'attributo id definisce il nome con cui l'oggetto può essere referenziato sia all'interno del file XML sia da codice Java. I tag property invece forzano il framework ad invocare il metodo setXXX identificato dall'attributo name, quindi al metodo setB viene passato l'oggetto che nel file XML è definto con id valorizzato a bb. Ricapitolando, il framework effettua per noi le seguenti operazioni:

```
B bb=new B();
A aa=new A();
aa.setB(bb);
```

Inoltre si può utilizzare l'ApplicationContext come una Hashtable e recuperare i vari oggetti usando come indice il valore dei vari attributi id presenti nei tag <bean>. Il primo vantaggio riscontrabile nell'utilizzo di questo fra-

REQUISITI

Conoscenze richieste
Java, SQL, XML, Velocity

Software
JDK 1.4, Spring 1.2.x e Eclipse 3.0

Impegno

Tempo di realizzazione

mework è che se si utilizzano in modo appropriato le interfacce si può cambiare il comportamento di una classe solo con poche modifiche ad un file XML.

LE BASI DI SPRING MVC

Spring MVC è la soluzione offerta da questo formidabile framework per la realizzazione di applicazioni web. Adottando il "solito" pattern IoC, sono state sviluppate dai suoi progettisiti delle API che, oltre a facilitare notevolmente l'approccio MVC, godono di un alto grado di apprendimento e di una flessibilità non riscontrabili in altri framework. Come al solito è prevista una particolare Servlet che funge da Dispatcher; ha quindi il compito di ricevere le richieste provenienti dal Browser e di smistarle al Controller previsto per quella determinata richiesta. Quindi il primo passo è quello di configurare il file di deploy web.xml per il mapping relativo alle richieste da passare alla nostra Servlet. Esso conterrà i seguenti tag XML:

```
<web-app>
  <servlet>
    <servlet-name>wall</servlet-name>
<servlet-class>org.springframework.web.servlet.
                      DispatcherServlet</servlet-class>
  </servlet>
  <servlet-mapping>
   <servlet-name>wall</servlet-name>
      <url-pattern>*.jhtm</url-pattern>
    </servlet-mapping>
</web-app>
```

Così facendo abbiamo configurato la nostra webapp in modo che tutte le richieste aventi estensione .jhtm vengano passate alla Servlet istanza della classe DispatcherServlet; a tale Servlet inoltre viene associato il nome logico wall. Ciò è molto importante in quanto concorre alla definizione del nome del file XML in cui risiedono le rimanenti configurazioni dell'applicazione. Infatti nella stessa directory WEB-INF dobbiamo inserire il file wall-servlet.xml che guida il framework nella definizione delle dipendenze tra i vari oggetti.

SPRINGWALL: UN ESEMPIO PRATICO

Prepariamoci adesso alla realizzazione di un applicativo web con le funzionalità molto simili a quelle di un normale guestbook (libro degli ospiti). Ogni utente può scrivere un messaggio nel libro degli ospiti per manifestare la propria opinione; la particolarità che vogliamo dare a questa applicazione è di natura grafica, infatti i messaggi saranno visualizzati su un muro virtuale. Per la realizzazione di tali funzionalità la nostra applicazione necessita di due Controller: uno per la visualizzazione dei messaggi ed uno per l'inserimento dei messaggi tramite un form HTML.

Ma come si implementa un Controller per il framework Spring MVC?

Il framework propone diverse soluzioni per la realizzazione di un Controller, analizziamo la più semplice che è anche quella che utilizzeremo per la visualizzazione dei messaggi. Spring definisce la seguente interfaccia:

```
public interface Controller{
            public ModelAndView
                handleRequest(HttpServletRequest req,
         HttpServletResponse res) throws Exception;
}
```

Quindi basta definire una classe che la implementi per ottenere oggetti funzionanti come Controller. Il metodo handleRequest restituisce un oggetto istanza della classe ModelAndView; come vedremo da qui a poco questi oggetti contengono informazioni inerenti la view da utilizzare ed i dati (model) da inserire nella view stessa.

```
public class WallController implements Controller{
   protected WallMessageDAO dao=null;
   protected int maxMsg=15;
   ….
   public ModelAndView
               handleRequest(HttpServletRequest req,
       HttpServletResponse res) throws Exception {
     Hashtable model=new Hashtable();
     ArrayList messages=this.dao.getMessages();
     messages=this.tryCut(messages);
     model.put("messages",messages);
```

I TUOI APPUNTI

COSA SERVE PER COMINCIARE

Per utilizzare Spring è necessario scaricare il framework presso il suo sito www.springframework.org. Una volta scaricato lo zip, bisogna scompattarlo in locale; sotto la directory dist sono presenti i jar di Spring che a secondo del contesto vanno copiati nel classpath dell'applicazione. Analogamente sotto la directory lib/velocity è presente il file velocity-1.4.jar, anch'esso è da includere nel classpath della nostra applicazione. C'è da notare che sotto docs\reference\pdf è presente il file spring-reference.pdf che contiene un guida completa al framework.

```
        return new
                ModelAndView("wall","model",model);
    }
    public void setDao(WallMessageDAO dao) {
        this.dao = dao;
    }
    public void setMaxDim(int dim){
        this.maxMsg=dim;
    }
}
```

La semantica della classe WallController è racchiusa tutta nel metodo handleRequest; per motivi di spazio non è stato utilizzato un layer di Business Logic. Quindi le operazioni inerenti la parte model del pattern MVC sono demandate al Controller stesso e ad un oggetto istanza di WallMessageDAO che ha il compito di interagire con la base dei dati. Infatti vengono recuperati tutti i messaggi dal DB, successivamente, se la dimensione dei messaggi supera la dimensione massima stabilita, vengono scartati i più vecchi con il metodo tryCut. A questo punto i dati che devono essere visualizzati dalla view vengono memorizzati in una Hashtable e successivamente vengono passati all'oggetto ModelAndView. Soffermiamoci su quest'ultimo passaggio perché merita un minimo di approfondimento: l'oggetto istanza di ModelAndView viene creato passando tre parametri nel costruttore; il primo identifica il nome della vista da utilizzare, il secondo invece il nome logico da utilizzare all'interno della vista per referenziare l'oggetto model (il terzo parametro) che contiene i dati che il Controller intende visualizzare.

INIETTIAMO LE DIPENDENZE

Diamo adesso un primo sguardo al file wall-servlet.xml. Come abbiamo già detto in precedenza, in questo file sono definite le dipendenze tra le varie entità che costituiscono il nostro progetto. In pratica Spring legge questo file ed in base alle "direttive" in esso contenute crea i vari oggetti e li lega tra di loro sfruttando i metodi setXXX che espongono.

```
<beans>
….. definizione di altri bean
<bean id="wallController"
        class="roby.springwall.client.WallController">
    <property name="dao" >
            <ref bean="wmdao" />
    </property>
    <property name="maxDim" >
            <value>15</value>
    </property>
</bean>
<bean id="urlMapping" class=
"org.springframework.web.servlet.handler.SimpleU
                                    rlandlerMapping">
    <property name="mappings">
      <props>
        <prop
            key="/wall.jhtm">wallController</prop>
        <prop
              key="/insert.jhtm">insertForm</prop>
      </props>
    </property>
</bean>
….. definizione di altri bean
</beans>
```

Il bean con id urlMapping serve a definire la strategia con cui il framework associa le richieste ai vari Controller. Nel nostro caso utilizziamo un SimpleUrlHandlerMapping

VALIDARE I DATI

Spring MVC prevede un meccanismo molto semplice e potente per la validazione dei dati provenienti da form HTML. Il framework definisce l'interfaccia Validator che espone due metodi supports e validate, per meglio comprendere il loro utilizzo analizziamo l'implementazione realizzata per la nostra applicazione.

```
public class InsertValidator implements Validator{
 public boolean supports(Class clazz) {
   return clazz.equals(WallMessageVO.class);
 }
  public void validate(Object obj, Errors errors) {
    WallMessageVO wmvo=(WallMessageVO)obj;
     if(wmvo.getMsg()==null || wmvo.getMsg().trim().equals("")){
       errors.rejectValue("msg",null,"Devi inserire un messaggio!");
     }
     if(wmvo.getSender()==null || wmvo.getSender().trim().equals("")){
        errors.rejectValue("sender",null,"Devi inserire il tuo nome!");
     }
  }
}
```

Il metodo supports può servire per identificare il contesto applicativo del Validator, nel nostro caso stiamo dicendo al framework che il nostro validator vuole trattare solo oggetti di tipo WallMessageVO. Il metodo validate invece stabilisce la modalità di validazione dei dati inviati. Il framework passa al validator un oggetto di tipo WallMessageVO e noi lo accettiamo, facciamo in modo che venga invocato il metodo onSubmit nel Controller associato, solo se i campi sender e msg non sono vuoti. In caso contrario viene utilizzato il metodo rejectValue presente nell'oggetto errors. Questo metodo pevede l'utilizzo di tre parametri, il primo definisce il nome del campo della form in cui visualizzare il messaggio di errore. Il secondo definisce il codice del messaggio di errore (per i nostri scopi non serve), mentre il terzo definisce il messaggio da visualizzare sulla form HTML (ne riparleremo quando studieremo la parte View).

che effettua un'associazione in base alla URL della richiesta. Alla URL che termina con /wall.jhtm associa il Controller identificato dal bean avente id valorizzato a wallController, mentre quello (non riportato nel precedente listato) con id valorizzato a insertForm si occuperà delle richieste che puntano a /insert.jhtm. Osservando il bean relativo a wallController ci accorgiamo che il framework effettua l'invocazione dei metodi setDao e setMaxDim, passando al primo il riferimento del bean wmdao e al secondo l'intero 15.

IL DATA LAYER

La nostra applicazione utilizza un semplice

wallmessage : Table

	Field Name	Data Type
🔑	id	AutoNumber
	msg	Memo
	sender	Text
▶	datet	Date/Time

Fig. 1: ***Struttura della tabella che memorizza i messaggi***

DataBase formato da una sola tabella Access avente la seguente struttura.
Il campo id è di tipo autonumber ed è la chiave primaria, msg memorizza i messaggi postati dagli utenti, sender invece memorizza l'autore del messaggio ed infine il campo datet serve a tenere traccia della data in cui è stato scritto il messaggio. Per il mapping della tabella utilizziamo il seguente Value Object:

```
public class WallMessageVO {
   protected int id=-1;
   protected String msg=null;
   protected String sender=null;
   protected GregorianCalendar date=null;
   … metodi get e set relativi alle
   … precedenti variabili
}
```

Per quanto riguarda l'interazione con il DB implementiamo un oggetto WallMessageDAO che espone i metodi per la gestione dei dati.

```
public class WallMessageDAO {
    protected JdbcTemplate jdbcTemplate=null;
    public void setJdbcTemplate(JdbcTemplate
                                  jdbcTemplate) {
       this.jdbcTemplate = jdbcTemplate;
    }
    public ArrayList getMessages(){
       ArrayList res=null,list=null;
       List li=this.jdbcTemplate.queryForList("select
*
             from wallmessage order by id desc");
       list= new ArrayList(li);
       Map map=null;
       WallMessageVO p=null;
       res=new ArrayList();
       for(int i=0;i<list.size();i++){
          map=(Map)list.get(i);
          p=new WallMessageVO();
       p.setId(((Integer)map.get("id")).intValue());
```

VELOCITY

Riprendiamo molto velocemente quanto visto nel numero 100 di ioProgrammo in cui abbiamo approfondito la conoscenza di questo splendido framework grazie all'articolo dell'ottimo Luigi Caputo. Velocity, progetto realizzato dalla Apache Foundation Group, è un template engine implementato interamente in Java; per template engine si intende una collezione di API che permette la generazione dinamica di contenuti testuali formattati secondo specifici modelli (template). In ambito web è possibile combinare insieme codice html/javascript con codice VTL. Il VTL è il linguaggio di scripting usato da Velocity, per scelta è molto semplice in quanto deve forzare chi lo utilizza ad effettuare le elaborazioni più complesse negli altri layer (probabilmente nella parte Model) dell'applicazione.
Gli elementi principali del linguaggio VTL sono i reference e le direttive; i primi sono preceduti del carattere '$', mentre le direttive sono introdotte dal carattere '#'
Ci sono tre tipi di reference: le variabili, le proprietà ed i metodi. Le variabili possono essere importate da codice Java oppure possono essere introdotte direttamente nel template; per effettuare quest'ultima operazione è necessario l'utilizzo della direttiva #set($v=<valore>).
Altre direttive degne di nota sono la direttiva di selezione #if(<condizione>) … #else …. #end e la direttiva di loop #foreach($item in $collection) … #end

```
          …. inserimento altri campi
          res.add(p);
       }
       return res;
    }
    public void deleteLess(int id){
       String dele="delete from wallmessagewhere
                                    id<="+id;
       this.jdbcTemplate.update(dele);
    }
    public void insert(WallMessageVO wm){
     String ins="insert into
          wallmessage(msg,sender,datet) values(";
                    ins+="'"+wm.getMsg()+"','"+
                           wm.getSender()+"','"
                    +wm.getFormattedDate()+"')";
                    this.jdbcTemplate.update(ins);
    }
}
```

Il metodo insert serve ad inserire un nuovo messaggio nel DB, mentre il metodo getMessages

Fig. 2: **Velocity combina la potenza di Java alla versilità dei template testuali**

serve a recuperare tutti i messaggi dal DB; infine deleteLess serve ad eliminare i messaggi più vecchi. E' importante notare come nel precedente codice non compaia nessun oggetto delle API JDBC. Infatti abbiamo voluto utilizzare la classe JdbcTemplate, questa facilita e riduce di molto il lavoro degli sviluppatori effettuando un wrapping delle principali funzionalità messe a disposizione dal package java.sql. Il metodo update consente di effettuare query di DELETE, INSERT ed UPDATE, mentre queryForList è utilizzabile per query di tipo SELECT in cui i risultati vengono restituiti in una java.util.List. Come possiamo notare in quest'ultimo caso non si ricorre all'utilizzo di un ResultSet, infatti queryForList restituisce una List di oggetti di tipo java.util.Map. Come per ResultSet ogni entry della tupla corrente può essere referenziata dal nome della colonna della tabella. Come al solito non ci resta che iniettare le dipendenze ricorrendo al file wall-servlet.xml.

```
<bean id="ds" class=
 "org.springframework.jdbc.datasource.DriverMana
                                  ge DataSource">
  <property name="driverClassName" >
  <value>sun.jdbc.odbc.JdbcOdbcDriver</value>
  </property>
   <property name="url" >
     <value>jdbc:odbc:WallMessage</value>
    </property>
</bean>
                               <bean id="jt"class="org.
             springframework.jdbc.core.JdbcTemplate>
   <property name="dataSource" >
        <ref bean="ds" />
   </property>
</bean>
<bean id="wmdao"
     class="roby.springwall.dao.WallMessageDAO">
   <property name="jdbcTemplate" >
       <ref bean="jt" />
   </property>
</bean>
```

Abbiamo visto in precedenza che il bean wmdao serve al WallController per interagire con il DB, a sua volta esso necessita di un oggetto istanza di JdbcTemplate che è definito dal bean avente id valorizzato a jt. Quest'ultimo utilizza un oggetto DriverManagerDataSource per connettersi al Data Base. Questa operazione viene effettuata per mezzo dei parametri driverClassName e url che identificano rispettivamente il driver JDBC utilizzato e l'URL di connessione.

INSERIAMO I DATI

E' giunto il momento di fornire al nostro sistema un modo per inserire i messaggi; a tal fine utilizzeremo un Controller un po' particolare. Infatti, così come avviene per altri framework, Spring MVC prevede un trattamento particolare per la gestione dei Form HTML. Come al solito noi utilizzeremo solo alcune delle API messe a disposizione dal framework; in particolare utilizzeremo un Validator per la validazione dei dati ed un Controller che lo utilizza.

```
public class InsertController extends
                                   SimpleFormController {
    protected WallMessageDAO dao=null;
    protected ModelAndView onSubmit(Objectobj){
       WallMessageVO wmvo=(WallMessageVO)obj;
       Hashtable model=new Hashtable();
       wmvo.setDate(new GregorianCalendar());
       this.dao.insert(wmvo);
       model.put("msg",wmvo);
       return newModelAndView
              (this.getSuccessView(),"model",model);
    }
    public void setDao(WallMessageDAO dao) {
           this.dao = dao;
    }
}
```

La prima cosa che possiamo notare è che questo Controller, contrariamente a quello visto in precedenza, non implementa direttamente l'interfaccia Controller ma lo fa indirettamente estendendo la classe SimpleFormController. Questa classe prevede diversi metodi onSubmit, nel nostro caso utilizziamo quello più adatto alle nostre esigue esigenze. Esso prevede l'utilizzo di un solo parametro che corrisponde all'oggetto che mappa i campi del Form HTML, tutto ciò ci viene "regalato" dal framework senza accedere direttamente ai parametri della richiesta http. Una volta ottenuto l'oggetto istanza della classe WallMessageVO, lo completiamo inserendo la data e successivamente inseriamo il nuovo oggetto sul DB. A questo punto non ci resta che restituire un oggetto ModelAndView opportunamente creato. Come possiamo notare il nome della vista da utilizzare (il primo parametro passa-

CONTROLLER E VARIABILI DI ISTANZA

Nell'implementare i Controller con Spring MVC bisogna rispettare alcune regole di buona programmazione; infatti, così come avviene per le Servlet, il framework istanzia un solo Controller (uno per ogni classe che implementa l'interfaccia Controller), quindi se si usano variabili (o oggetti) istanze di Classe si corre il rischio che diverse richieste http accedano contemporaneamente, e quindi in regime di concorrenza, a tali variabili.

to nel costruttore) lo ricaviamo attraverso il metodo getSuccessView, esso utilizza un parametro definito nel file wall-servlet.xml. Infatti non dobbiamo mai dimenticare che con Spring il comportamento degli oggetti definiti tramite codice Java dipende fortemente da quanto stabilito nei file di configurazione.

```
<bean id="insertValidator"
    class="roby.springwall.client.InsertValidator"/>
 <bean id="insertForm"
    class="roby.springwall.client.InsertController">
   <property name="dao" ><ref bean="wmdao"
                                    /></property>
   <property
  name="commandName"><value>wmsg</value>
                                      <property>
    <property name="commandClass">
         <value>roby.springwall.vo.WallMessageVO
                                        </value>
                                    </property>
     <property name="validator">
<ref bean="insertValidator"/></property>
     <property
         name="formView"><value>insert</value>
                                    </property>
     <property
   name="successView"><value>okInsert</value>
                                    </property>
</bean>
```

Analizzando questo frammento del file wall-servlet.xml ci accorgiamo che la costruzione del Controller in questione è leggermente più complicata di quella vista in precedenza. Finalmente possiamo constatare che la vista utilizzata in caso di successo è quella definita dalla property avente name valorizzato a successView; anche in questo caso il framework inserisce tramite il metodo setDao l'oggetto istanza di WallMessageDAO e definito nel bean avente come id il valore wmdao. La property formView specifica il link al Form HTML utilizzata per l'inserimento dei dati. Le property commandName e commandClass concorrono alla definizione dell'oggetto che effettua il binding dei valori inseriti dall'utente: il primo specifica il nome logico utilizzato nella vista (lo capiremo meglio da qui a poco), mentre il secondo definisce il nome della classe da utilizzare per istanziare il precedente oggetto. La property validator serve a definire il Validator da utilizzare per la validazione della form.

VIEW CON VELOCITY

Finora ci siamo occupati dei Controller e della gestione dei dati, per quanto riguarda la loro rappresentazione sul web ci siamo solo limitati a sottolineare che i Controller, una volta elaborati i dati, ne estraggono un modello (Model) che li rappresenta e lo affidano alla parte View dell'applicazione. Uno dei punti di forza di Spring MVC deriva proprio da questo comportamento, infatti finora non abbiamo fatto nessuna supposizione

Fig. 3: **I messagi scritti sul muro**

sulla tecologia da utilizzare nella nostra applicazione. Questa netta separazione tra i vari layer consente di scegliere la tecnologia che più ci piace; per la realizzazione del nostro muro utilizzeremo Velocity, è utile sottolineare però che, senza cambiare una solo riga di codice Java, avremmo potuto optare per le JSP.E' ovvio che tale scelta influisce solo sulla definizione del file wall-servlet.xml.

```
<bean id="velocityConfig" class=
        "org.springframework.web.servlet.view.velocity.
                                   VelocityConfigurer">
<property name="resourceLoaderPath">
             <value></value>
</property>
</bean>
```

Questo bean serve per l'inizializzazione di Velocity. Infatti tramite la property resourceLoaderPath si stabilisce il root path in cui andare a cercare i file .vm che rappresentano i template Velocity. Quella che abbiamo appena visto è una configurazione specifica di Velocity che nel caso delle JSP non va utilizzata, invece quello di cui necessita in tutti i casi una webapp sviluppata con Spring MVC è la definizione di un bean con id valorizzato a viewResolver.

```
<bean id="viewResolver" class=
        "org.springframework.web.servlet.view.velocity.
                                VelocityViewResolver">
 <property name="prefix">
<value>/WEB-INF/vm/</value></property>
<property name="suffix"><value>.vm</value>
</property>
```

SETTERS O COSTRUTTORI?

Spring supporta due modi per effettuare la Dependency Injection (o Inversion of Control) tra gli oggetti di un'applicazione:

- ***setter-based: è la modalità in cui i bean sono legati tra di loro utilzzando i metodi set<Proprieta>. Questa modalità è quella utilizzata in questo articolo e suggerita nella documentazione stessa.***
- ***constructor-based: è la modalità in cui i bean sono legati tra di loro sfruttando il passaggio degli oggetti nei costruttori***

```
<propert
name="exposeSpringMacroHelpers"><value>true<
                                    /value></property>
</bean>
```

Il framework cerca nel file XML un bean avente questa caratteristica e grazie ad esso capisce quale tecnologia utilizzare per il rendering delle view. Nel nostro caso stiamo definendo un prefisso ed un suffisso per l'identificazione della view; per esempio

Fig. 4: *I messaggi di errore notificati dal Validator*

nel WallMessageController abbiamo visto che l'oggetto ModelAndView veniva costruito con il parametro relativo alla view valorizzato con la stringa "wall". In realtà quel valore composto con il prefisso ed il suffisso identifica la vista contenuta nel file: <webapp>/WEB-INF/vm/wall.vm

La parte saliente di questo file è riassunto nel seguente codice:

```
<table ... background="images/brick.jpg">
   #foreach($m in $model.messages)
      <tr><td> $m.formattedDate </td></tr>
      <tr><td> $m.msg  </td></tr>
      <tr><td> $m.sender </td></tr>
    #end
 </table>
```

La direttiva #foreach ci consente di scorrere tutti gli elementi $m contenuti in $model.messages, ognuno di questi elementi rappresenta una istanza della classe WallMessageVO e quindi tramite i campi formattedDate, msg e sender riusciamo a visulaizzare le informazioni che contiene. Vediamo adesso di analizzare la parte più importante della vista insert.vm:

```
<form method="POST" action="">
   <table width="90%" border="0" cellspacing="0"
                                        cellpadding="0">
   <tr><td>
    #springBind("wmsg.sender")
                          $status.errorMessage <br/>
    autore <input type="text" name="sender"
                               value="$!status.value">
    </td></tr><tr><td>
    #springBind("wmsg.msg")
                           $status.errorMessage<br/>
    messaggio<br/>  <textarea name="msg"
   rows="8"     cols="40">$!status.value</textarea>
   </td></tr><tr><td>
   <input type="submit" value="Invia"
                                      name="submit"/>
   </td></tr>
   </table>
</form>
```

Poco prima del primo tag input notiamo il seguente codice VTL: #springBind("wmsg.sender") $status.errorMessage

La prima è una direttiva che serve a dichiarare al framework che il successivo input (di tipo text) serve a valorizzare il campo sender dell'oggetto wmsg, la scelta del nome dell'oggetto ed il tipo sono fissati nel file xml dalle property commandName e commandClass del bean avente id valorizzato a InsertController. La successiva espressione VTL $status.errorMessage gioca un ruolo attivo in caso di mancata compilazione dell'input in questione; infatti, come avevamo già visto nella classe InsertValidator, esso determina la visualizzazione del messaggio di errore "Devi inserire il tuo nome!". Un comportamento del tutto simile viene assunto dal succesivo tag <textarea>.

LA CONFIGURAZIONE DEL DB

Questo progetto utilizza come database un DB Access memorizzato nel file wallmessage.mdb presente nel cd nella sottodirectory WEB_INF della webapp. Per configurare il database Access è necessario:

- **Eseguire Data Sources (ODBC) presente in Settings/Control Panel/ Administrative Tools**
- **Scegliere User DSN/Add, selezionare Microsoft Access Driver(*.mdb) e poi il pulsante Finish**
- **In Data Source Name inserire la stringa WallMessage**
- **Nel riquadro Database, con il pulsante Select... selezionare il file .mdb in cui risiede il DB**
- **Confermare con OK nelle due form aperte**

CONCLUSIONI

In questo articolo abbiamo visto ed apprezzato le caratteristiche principali di Spring MVC, abbiamo imparato ad utilizzarlo insieme a Velocity sottolinenado che il codice scritto non preclude l'utilizzo di altre tecnologie come le JSP. A questo punto non ci resta che approfondirne la conoscenza restando sempre sintonizzati con il framework Spring che sta deliziando sempre più il mondo degli sviluppatori Java.

Roberto Sidoti

L'AUTORE

Roberto Sidoti è ingegnere informatico, attaulmente si occupa di servizi VoIP basati sul protocollo SIP presso Telecom Italia LAB come consulente esterno. I suoi maggiori interessi riguardano le applicazioni distribuite sviluppate con tecnologia J2EE, da marzo 2005 si occupa dello sviluppo e della gestione del progetto GeiNuke (www.hostingjava.it/-geinuke/).

CREARE L'HELP DI UN'APPLICAZIONE

IL FRAMEWORK .NET FORNISCE LE CLASSI NECESSARIE AD IMPLEMENTARE SISTEMI DI HELP ON-LINE PER LE APPLICAZIONI WINDOWS FORMS. ECCO COME UTILIZZARLE PER DOTARE OGNI APPLICAZIONE DEL SUPPORTO CHE GLI UTENTI SI ASPETTANO

Scrivere la documentazione ed i file di help di un'applicazione è spesso un compito sottovalutato, o addirittura tralasciato. Magari perché ritenuto meno nobile dello sviluppo, oppure perché noi sviluppatori crediamo di non poter togliere tempo alla programmazione vera e propria.
Ma chi dovrebbe scrivere il manuale e la guida, se non coloro i quali hanno creato l'applicazione? Affidarla a terzi rallenterebbe o peggiorerebbe il processo; dato che prima questi dovrebbero acquisire conoscenza approfondita dell'applicazione, probabilmente, facendo perdere tempo prezioso allo sviluppatore, a causa di continue interruzioni per maggiori informazioni sul funzionamento.
Senza un sistema di Help, o di un manuale decente, la vostra applicazione, che magari è ottima, probabilmente la migliore nello svolgere il suo compito, potrebbe risultare inutilizzabile per l'utente medio, abbassando così di molto il suo valore, ed il suo appeal per gli utenti, che magari preferiranno pagare un altro tool, pur di avere un help a cui rivolgersi quando non sanno come utilizzare una funzionalità più avanzata, un pulsante, un menu mai visto.

TECNICHE DI AIUTO

Esistono diverse tecniche per aggiungere una funzionalità di help ad un'applicazione. La maggior parte sono ormai diventate uno standard de facto sia nel mondo Windows sia Unix.
Ad esempio, praticamente, ogni applicazione ha un menu Help, spesso abbreviato con un punto interrogativo, oppure in ognuna il tasto F1 attiva un help contestuale sulla funzione che si sta utilizzando, ed ancora molte utilizzano dei fumetti o tooltip, che appaiono posizionando il mouse su un controllo, e spiegando la sua funzione.

Tutti sono sistemi semplici da implementare anche nel mondo delle applicazioni .NET, il cui framework fornisce le classi necessarie per realizzarle.
In questo articolo creeremo dunque una semplice applicazione, e vedremo come aggiungere le varie modalità di aiuto in linea.

L'APPLICAZIONE SENZA HELP

Creiamo intanto una semplice applicazione Windows costituita da una singola finestra, e da qualche controllo, come pulsanti, textbox e così via.
Dato che lo scopo dell'articolo non è quello di mostrare la creazione di un'applicazione mostriamo solo l'interfaccia che utilizzeremo

Fig.1: *Un'applicazione a cui aggiungere l'help*

nel prosieguo, rinviando al codice allegato nel CD o sul sito Web di ioProgrammo per chi non volesse crearla in modo autonomo.
La nostra applicazione si compone semplicemente di una Form per inserire i dati personali, ad esempio per una rubrica, o per un sistema di gestione clienti. Inizialmente l'applicazione non avrà alcun sistema che aiuti l'utente a capire come inserire i dati, oppure che lo avverta se i dati inseriti non sono validi, pensiamo ad esempio al CAP, che deve rispettare un certo formato numerico.

TOOLTIP E STATUSBAR

Supponiamo che la la nostra form possieda un pulsante con l'etichetta Carica, ma a che serve? Utilizzando un tooltip quando si posiziona il puntatore del mouse su di esso possiamo far apparire una descrizione od una spiegazione.
Con Visual Studio è sufficiente trascinare dalla casella dei controlli il componente ToolTip sulla Form per fare in modo che ogni controllo sia dotato di una nuova proprietà, Tooltip on Nometooltip.
Impostando una stringa essa sarà utilizzata come tooltip per il controllo. La stessa cosa si può ottenere naturalmente via codice.
Per creare un'istanza del ToolTip è necessario invocare il costruttore di default, e poi utilizzare il metodo SetTooltip specificando il controllo e la stringa da mostrare.

```
ToolTip mioTooltip = new ToolTip();
mioTooltip.SetToolTip(btCarica, "Carica la foto di
                                    questo cliente");
 mioTooltip.SetToolTip(btClose, "Chiudi la
                                         finestra");
```

Una sola istanza di ToolTip può dunque essere utilizzata con tutti i controlli presenti sulla form.
La figura 2 mostra il tooltip che appare posizionando il puntatore del mouse sul pulsante Carica.

Fig.2: ***Tooltip per un pulsante***

Un altro sistema per mostrare informazioni su ciò che sta accadendo su una finestra dell'applicazione, è quello di mostrare una stringa sulla barra di stato, che varia ad esempio spostando il focus da un controllo all'altro, mostrando dunque in tempo reale una descrizione dell'informazione data dal controllo. Per realizzare questo effetto, dopo aver aggiunto un controllo StatusStrip, e creata una ToolStripStatusLabel al suo interno, sarà necessario gestire l'evento MouseMove dei controlli interessati e dell'intera Form, impostando la proprietà Text della ToolStrip StatusLabel, ad esempio così:

```
    private void txtIndirizzo_MouseMove(object
                         sender,MouseEventArgs e)
{
        this.toolStripStatusLabel.Text =
                         "L'indirizzo del cliente";
}
private void txtCittà_MouseMove(object sender,
                              MouseEventArgs e)
{
        this.toolStripStatusLabel.Text =
                 "La città di residenza del cliente";
}
private void cboCAP_MouseMove(object sender,
                              MouseEventArgs e)
{
        this.toolStripStatusLabel.Text = "Il CAP
                        di residenza del cliente";
}
private void Form1_MouseMove(object sender,
                              MouseEventArgs e)
{
        this.toolStripStatusLabel.Text = "";
}
```

Notate come nel caso dell'evento MouseMove relativo all'intera Form, resettiamo il contenuto con una stringa vuota.

LA CLASSE ERRORPROVIDER

La classe ErrorProvider permette di dare all'utente dell'applicazione un feedback su determinate situazioni di errore. Classicamente essa viene utilizzata quando dei controlli da valorizzare vengono riempiti con valori non validi. Pensate ad esempio ad una TextBox che deve contenere solo numeri, oppure che deve essere validata secondo certi criteri, come numeri solo positivi, o stringhe in un certo formato. Mediante i metodi della classe ErrorProvider, è possibile mostrare all'utente un'icona al lato di un determinato controllo che non è stato valorizzato correttamente.
Supponiamo ad esempio che il nome del cliente sia un campo obbligatorio. Quando sia valida la Form, per esempio chiudendola, è possibile verificare la presenza del nome del

I TUOI APPUNTI

I COMPONENTI PROVIDER

Esistono dei componenti forniti dal .NET Framework, che pur non essendo visibili all'utente permettono di dotare i controlli grafici di nuove proprietà. Tali componenti sono detti appunto extender provider, e comprendono i tre visti nell'articolo: Tooltip, ErrorProvider ed HelpProvider. Gli sviluppatori possono creare il proprio provider implementando l'interfaccia IExtenderProvider.

cliente e se il campo è vuoto mostrare un'icona di errore:

```
private void btClose_Click(object sender,
                                        EventArgse)
{
    if (txtNome.Text == String.Empty)
    {
        ErrorProvider error = new
                                ErrorProvider(this);
        error.SetError(txtNome, "Inserire il nome");
    }
}
```

L'icona mostra un punto esclamativo, per default sulla destra del controllo interessato, che lampeggia un determinato numero di volte, e che mostra un ToolTip posizionandovi sopra il puntatore, come mostrato in figura 3.

Fig.3: *L'icona di errore associata ad una TextBox*

È possibile configurare diversi aspetti dell'ErrorProvider, per mezzo di proprietà e metodi.
Impostando la proprietà BlinkRate è possibile variare la velocità di lampeggio dell'icona di errore, data in millisecondi, e che per default è di 250 millisecondi. Se invece, ad esempio, vogliamo che l'icona lampeggi ad intervalli di 1 secondo, sarà sufficiente impostare il valore di BlinkRate a1000:

```
error.BlinkRate=1000;
```

La proprietà BlinkStyle invece permette di definire quando l'icona deve lampeggiare, usando uno dei valori dell'enumerazione ErrorBlinkStyle. Per default il blink è sempre attivo, quindi impostata a AlwaysBlink, e quindi esso si verifica ad ogni occorrenza dell'errore.
Se invece si desidera che l'icona rimanga fissa, si può usare il valore NeverBlink, o ancora se il lampeggio deve verificarsi solo su errori diversi da eventuali errori precedenti sullo stesso controllo, bisogna utilizzare il valore BlinkIfDifferentError.

```
error.BlinkStyle = ErrorBlinkStyle.NeverBlink;
```

È possibile anche configurare il posizionamento dell'icona stessa oppure un'icona personalizzata al posto del punto esclamativo.
Il metodo SetIconAlignment imposta l'allineamento dell'icona rispetto al controllo associato, utilizzando uno dei valori dell'enumerazione ErrorIconAlignment. Ad esempio se si vuol posizionare l'icona alla sinistra della TextBox, ed allineata al centro di essa scriveremo:

```
error.SetIconAlignment(txtNome,
                    ErrorIconAlignment.MiddleLeft);
```

La distanza dal controllo è invece controllata mediante il metodo SetIconPadding, con cui è possibile specificare il numero di pixel da utilizzare per il distanziamento:

```
error.SetIconPadding(txtNome, 1); //1 solo pixel
                                        di padding
```

Per utilizzare un'icona personalizzata invece, basta impostare la proprietà Icon, creandone ad esempio una da un file con estensione ico, così:

```
error.Icon=new Icon("erroricon.ico");
```

Ciò senza dubbio darà un aspetto personale alla vostra applicazione, come ad esempio quella in figura 4.

Fig.4: *Un'icona di errore personalizzata*

LA CLASSE HELPPROVIDER

L'help contestuale consente di fornire all'utente aiuto ed informazioni su un particolare controllo, semplicemente selezionandolo e utilizzando i tasti di scorciatoia opportuni. Per far ciò, è possibile in .NET utilizzare la classe HelpProvider. Una volta aggiunto un componente HelpProvider, tramite la casella degli strumenti di Visual Studio oppure via codice, sarà possibile impostare la stringa di help per ogni controllo che si vuole, ad esempio, per la TextBox txtNome scriveremo:

```
HelpProvider helpProvider = new HelpProvider();
helpProvider.SetHelpString(txtNome, "Il nome del
                                          cliente");
helpProvider.SetHelpString(btCarica, "Carica un
        file immagine che rappresenta la foto del 0
                                          cliente");
helpProvider.SetHelpString(btClose, "Chiude la
                                          finestra");
```

Cliccando sull'HelpButton della Form, il puntatore visualizzerà un punto interrogativo, ed a questo punto ogni controllo per cui è stata impostata una stringa di Help visualizzerà al click su di esso tale stringa in una sorta di tooltip, come mostrato in figura 5.

Fig.5: ***Help contestuale del pulsante***

La stessa cosa avverrà, senza alcuna aggiunta di codice, cliccando il tasto F1 dopo aver impostato il focus su un controllo.
Per visualizzare un pulsante di Help sulla finestra, innanzitutto è necessario impostare a true la proprietà HelpButton della nostra Form, e quindi impostare a false le proprietà MaximizeBox e MinimizeBox.
Ciò visualizzerà sulla barra del titolo della finestra un pulsante con un punto interrogativo, che una volta cliccato attiverà l'help contestuale, situazione verificabile dal fatto che anche il puntatore del mouse cambierà
Se avete utilizzato Visual Studio per aggiungere il controllo HelpProvider, ogni controllo sarà dotato di una nuova proprietà HelpString on NomeHelpProvider nella finestra delle proprietà, da impostare con la stringa di Help per il controllo stesso.

UTILIZZARE UN FILE DI HELP

Sempre tramite la classe HelpProvider è possibile aprire un file di help esterno, sia esso un file html, che un file chm, cioè il classico file di help HTML compilato, con sommario, indice, ricerca.
Per impostare il file HTML da aprire, basta impostare la proprietà HelpNamespace utilizzando il percorso del file, ed invocare il metodo SetShowHelp per indicare che l'help è attivo su un determinato controllo.

```
HelpProvider htmlHelpProvider = new
                                HelpProvider();
htmlHelpProvider.SetShowHelp(btReadMe, true);
htmlHelpProvider.HelpNamespace =
                                "readme.htm";
```

Ricordate che se viene impostata la proprietà HelpNamespace allora la HelpString sarà ignorata e non verrà più visualizzata come Tooltip premendo il tasto F1 o utilizzando l'HelpButton della Form, ma al suo posto verrà aperto il file di guida stesso.
La HelpString sarà comunque accessibile per ogni evenienza invocando il metodo GetHelpString.
Più interessante senza dubbio l'utilizzo dei file CHM, in quanto è possibile associare ad ogni controllo dell'interfaccia una keyword, e quindi aprire la guida selezionando proprio la pagina che descrive la funzionalità desiderata.
Per realizzare un file di guida CHM potete utilizzare sia tool commerciali, che il gratuito HTML Help Workshop di Microsoft. Non ci soffermiamo qui sulla realizzazione di un help, cosa che esula dai nostri scopi, ma utilizzeremo un file chm qualunque. D'altronde, data la diffusione del formato ne trovate a decine in ogni installazione di Windows.
Prendiamo ad esempio il file di guida di Notepad (notepad.chm), che dovreste ritrovarvi anche voi nella directory C:\Windows \Help.
Se la proprietà HelpNamespace è stata impostata ad un percorso di un file di help, quest'ultimo verrà visualizzato usando i parametri che devono essere impostati mediante il metodo SetHelpNavigator ad uno dei valori

MICROSOFT HTML HELP

Per creare una guida in linea potete usare HTML Help Workshop, se non lo possedete, è scaricabile gratuitamente da microsoft.com, esattamente all'url http://go.microsoft.com/fwlink/?LinkId=14188, insieme alla relativa documentazione che invece trovate all'url http://go.microsoft.com/fwlink/?LinkId=14581.

dell'enumerazione HelpNavigator, riassunti nell'elenco di seguito:

- AssociateIndex: apre l'help file alla prima lettera dell'argomento specificato.
- Find: apre l'help alla pagina di ricerca
- Index: apre l'help alla pagine dell'indice
- KeywordIndex:apre l'help all'argomento dell'indice specificato se esiste, oppure a quello più vicino.
- TableOfContents: apre il sommario dell'help
- Topic: apre lo specifico argomento se esiste.
- TopicId: apre l'argomento identificato dal numero specificato.

Supponiamo ad esempio di volere, alla pressione del tasto F1 con il focus su un dato controllo, aprire il file notepad.chm, selezionando l'indice della guida stessa, ed esattamente in corrispondenza della parola chiave "testo", se esiste fra le keyword disponibili. In questo caso utilizzeremo i metodi SetHelpNavigator e SetHelpKeyword:

```
HelpProvider chmHelp = new HelpProvider();
chmHelp.HelpNamespace = "notepad.chm";
chmHelp.SetHelpNavigator(btHelpIndex,
                              HelpNavigator.Index);
chmHelp.SetHelpKeyword(btHelpIndex,"ricerca");
```

Se si vuol aprire la guida posizionandosi ad un determinato sotto argomento, bisogna fornire il percorso dell'argomento nel file, che potete ottenere cliccando con il tasto destro, nella pagine dell'argomento stesso e poi cliccando su proprietà, quindi bisognerà utilizzare come HelpNavigator il parametro Topic, ad esempio:

```
chmHelp.SetHelpNavigator(btHelpIndex,
                              HelpNavigator.Topic);
chmHelp.SetHelpKeyword(
                         btHelpIndex,"notepad.chm
                  :/win_notepad_utf_description.htm");
```

Infine vediamo come aprire l'indice posizionandosi alla prima parola che inizia con una determinata lettera. È necessario in questo caso impostare il valore HelpNavigator. AssociateIndex e quindi la lettere che interessa:

```
chmHelp.SetHelpNavigator(btHelpIndex,
                         HelpNavigator.AssociateIndex);
chmHelp.SetHelpKeyword(btHelpIndex, "a");
```

La classe HelpProvider utilizza internamente i metodi della classe Help, che fornisce allo sviluppatore solo metodi statici pubblici. La classe Help è naturalmente utilizzabile direttamente nel vostro codice.
Vediamo come utilizzarla ad esempio per creare il classico menu di Help, con le voci Sommario, Indice e Cerca, e aprire il file guida posizionandolo proprio ad una delle tre voci. Per ottenere lo scopo, basta invocare il metodo ShowHelp, con l'adeguato HelpNavigator ed eventualmente con una chiave di ricerca:

```
private void sommarioToolStripMenuItem_Click
                       (object sender, EventArgs e)
{
    Help.ShowHelp(this, "notepad.chm",
                   HelpNavigator.TableOfContents);
 }
private void indiceToolStripMenuItem_Click(object
                              sender, EventArgs e)
{
    Help.ShowHelp(this, "notepad.chm",
                              HelpNavigator.Index);
}
private void cercaToolStripMenuItem_Click(object
                              sender, EventArgs e)
{
    Help.ShowHelp(this, "notepad.chm",
                   HelpNavigator.Find,"keyword");
}
```

CONCLUSIONI

Abbiamo mostrato come rendere le applicazioni più user friendly, dotandole di un help in linea. Ciò è stato possibile utilizzando le classi ed i controlli previsti per tale scopo dal .NET framework, utilizzando Tooltip, StatusBar, ErrorProvider ed infine Help Provider per aprire file di help esterni.
I metodi utilizzabili sono davvero tanti, e corredare un'applicazione di una serie di aiuti che ne migliorino l'usabilità, attribuisce al software un valore davvero elevato. Inoltre è molto probabile che risparmierete molto tempo in formazione e assistenza. Non vi resta che iniziare a pensare a dotare anche le vostre applicazioni di un help.

Antonio Pelleriti

L'AUTORE

Antonio Pelleriti, ingegnere informatico, si occupa di .NET sin dalle prime versioni Beta. Potete contattare l'autore per suggerimenti, critiche o chiarimenti all'indirizzo e-mail antonio.pelleriti@ioprogrammo.it, sul forum di ioProgrammo all'url forum.ioprogrammo.net o sul sito web www.dotnetarchitects.it

FILTRI GRAFICI IN JAVA

QUESTO ARTICOLO ILLUSTRA COME CARICARE UN'IMMAGINE, VISUALIZZARLA IN UNA FINESTRA E APPLICARLE DIVERSI FILTRI PER OTTENERE EFFETTI GRAFICI INTERESSANTI, COME MARMO, OLIO O CRISTALLIZZAZIONE.

Con Java fare grafica avanzata è possibile? Certamente sì. In questo articolo mostreremo come creare un'applicazione in grado di applicare una serie di filtri su immagini, né più né meno di quanto fa ad esempio Photoshop. Per i nostri scopi ci serviremo di una serie di librerie liberamente scaricabili dal sito ttp://www.jhlabs.com/. Il nostro scopo sarà didattico. Quindi in un primo momento creeremo un'interfaccia base, dalle quale saremo in grado di derivare tante applicazioni specializzate ciascuna su un unico filtro. Perciò armiamoci di compilatore Java e di un buon IDE e iniziamo!

CHE VUOL DIRE JHLABS?

JH Labs è un sinonimo di Jerry Huxtable, un esperto sviluppatore Java che ha dedicato molte energie alla grafica e alla manipolazione delle immagini sulla piattaforma Java. Sul suo sito è possibile trovare diversi package Open Source e interessanti applicazioni, oltre ad alcune Applet che possono essere provate online. Tra le applicazioni si trovano un editor di immagini, che utilizza il package di filtri che viene illustrato nell'articolo, un visualizzatore di mappe, alcuni Layout Manager, un visualizzatore di font Unicode, alcuni giochi e altre cose interessanti, come un'Applet per creare animazioni di particelle, per intenderci quelle utilizzate per simulare fiamme o fuochi d'artificio e utilizzati anche negli effetti speciali cinematografici.

UNA CLASSE PER I TEST

Per provare il funzionamento dei filtri grafici, di cui parliamo in questo articolo, creeremo una classe di prova che fungerà da contenitore di tutto il codice che farà da supporto necessario alle operazioni di filtraggio. Questa classe si chiama ImageFilterTestBed ed è dichiarata astratta. Nelle sottoclassi inseriremo il codice dei filtri da cui passerà l'immagine prima di essere visualizzata.
La visualizzazione si baserà su un normale JFrame a cui assoceremo un Layout Manager BorderLayout. La presentazione dell'immagine sarà affidata a un oggetto JLabel, che in Swing implementa una semplice etichetta di testo.
Alla JLabel verrà anche associato un bordo, in modo da ottenere un simpatico riquadro che renda l'idea di una foto stampata.
La dichiarazione della classe è la seguente:

```
/**
 * ImageFilterTestBed
 */

package it.edmaster.ioprogrammo.filtrigrafici;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.IOException;
import javax.imageio.ImageIO;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.border.Border;

/**
 * Classe generica per la prova dei filtri. Produce
 * una rappresentazione dell'immagine basata su una
 * finestra JFrame
 *
 * @author mbigatti
 */

public abstract class ImageFilterTestBed {
```

l'unico attributo è un oggetto di tipo JFrame, che rappresenta la finestra in cui è visualizzata l'immagine:

```
        /** finestra */
        JFrame frame;
```

il costruttore non fa altro che richiamare il metodo createUI(), che si occupa di creare l'interfaccia utente:

```
/**
 * Crea la finestra
 * @throws IOException
 */
public ImageFilterTestBed() throws
                        IOException {
        createUI();
}
```

Il metodo createUI()

Come già detto, si tratta del metodo che viene richiamato nel costruttore e serve a creare l'interfaccia utente. Vediamolo nel dettaglio. La prima operazione che eseguirà sarà leggere dal CLASSPATH un'immagine. Lo farà attraverso le API Image I/O. Nel nostro esempio l'immagine si chiama IMG_5950.jpg e per caricarla si ottiene per prima cosa un URL che fa riferimento alla posizione fisica del file attraverso il metodo getResource():

```
/**
 * Crea l'interfaccia utente
 * @throws IOException
 */

void createUI() throws IOException {
        //carica l'immagine dal classpath
        BufferedImage image =
    ImageIO.read(getClass().getResource
                ("/IMG_5950.jg"));
```

l'immagine viene poi passata al metodo filter(), che ne esegue la manipolazione. Il risultato viene passato ad un nuovo oggetto ImageIcon, attraverso il metodo setImage():

```
//esegue il filtro e crea una icona da
//associare all'etichetta

image = filter((BufferedImage) image);
ImageIcon icon = new ImageIcon();
        icon.setImage(image);
```

in seguito viene creata l'etichetta e impostato un bordo costruito nel metodo create Border(), che verrà illustrato in seguito:

```
//crea l'etichetta e aggiunge il bordo

JLabel label = new JLabel(icon);
label.setBorder( createBorder() );
```

il passaggio finale consiste nell'istanziazione di un oggetto JFrame, nel cui titolo viene passato il nome del filtro. Questo è ottenuto dal metodo getFilterName(). Viene poi tolta la possibilità di ridimensionare la finestra. Viene impostato il Layout Manager BorderLayout. Poi viene aggiunta l'etichetta. La chiamata al metodo pack() ridimensiona la finestra in modo che la dimensione sia la minore possibile. La chiamata a setVisible() visualizza la finestra:

```
//crea la finestra, aggiunge l'immagine e  la
                        visualizza
frame = new JFrame("Filtro " +
                getFilterName());
frame.setResizable(false);
frame.getContentPane().setLayout( new
                BorderLayout() );
```

DIFFERENZE FRA AWT E SWING

Diversamente da AWT, in Swing le etichette possono includere anche immagini. In questo caso si parla di icone, definite da oggetti di interfaccia Icon. Una classe concreta che implementa Icon è ImageIcon, a cui è possibile passare un oggetto di tipo Image.

```
        frame.getContentPane().add(label);
        frame.pack();
        frame.setVisible(true);
}
```

Creare il bordo dell'etichetta è un'operazione leggermente complessa, perché sfrutteremo la combinazione di diversi bordi. Il codice relativo è scritto nel metodo createBorder(). Per prima cosa viene creato un bordo esterno di 20 pixel su ciascun lato, assegnato alla variabile space. Poi viene creato un bordo bianco di 15 pixel che verrà utilizzato per contornare la fotografia. Poi vengono creati due bordi grigi di intensità diversa, che vengono combinati in modo tale da ottenere un effetto di tridimensionalità. Questo bordo combinato viene ottenuto con il metodo create CompoundBorder(). Il bordo grigio viene poi combinato con quello bianco, e il risultato viene unito con il bordo esterno:

```
/**
 * Crea il bordo della fotografia,
 * combinando diversi bordi
 * @return
 */

Border createBorder() {
        //spazio esterno
        Border space =
        BorderFactory.
        createEmptyBorder(20,20,20,20);
```

```
            //interno bianco
            Border mu =
BorderFactory.createLineBorder(Color.WHITE,15);
            //lineee
            Border b1 =
BorderFactory.createMatteBorder(1,1,0,0,Color.LIGH
                                              _GRAY);
            Border b2 =
BorderFactory.createMatteBorder(0,0,1,1,Color.GRAY;
            Border r1 =
BorderFactory.createCompoundBorder(b1, b2);
            Border r2 =
BorderFactory.createCompoundBorder(r1, mu);
            Border r3 =
BorderFactory.createCompoundBorder(space, r2);
            return r3;
      }
```

La classe si conclude con due metodi astratti, che dovranno essere ridefiniti nelle sottoclassi. Il metodo filter() si aspetta un parametro di tipo BufferedImage e ritorna un valore dello stesso tipo. Questo metodo esegue la modifica dell'immagine. L'altro metodo astratto è getFilterName(), che ritorna il nome del filtro. Questa è una semplice descrizione utilizzata nel titolo della finestra.

```
      /**
       * Esegue la manipolazione dell'immagine
       * @param image
       * @return
       */

      abstract BufferedImage filter(BufferedImage
                                              image);
      /**
       * Ritorna il nome del filtro da visualizzare
       * nel titolo della finestra
       * @return
       */
      abstract String getFilterName();
}
```

UN PROGRAMMA DI PROVA

Quello che faremo ora sarà creare una classe derivata da ImageFilterTestBed, che però non utilizzerà alcun filtro. L'immagine verrà quindi presentata senza modifiche, così come è salvata nel file di prova. Questa classe di prova si chiamerà NoFilter, ed estenderà la ImageFilterTestBed. Essendo una classe concreta, deve implementare i metodi filter() e getFilterName(). Il primo metodo non farà altro che ritornare il parametro, senza modificarlo. In questo modo l'immagine da visualizzare sarà la medesima di quella in ingresso. Il metodo getFilterName() ritornerà semplicemente il valore "Neutro". Oltre a questi due metodi saranno presenti un costruttore e il metodo main(). Il primo non farà altro che creare il costruttore della superclasse, che esegue la creazione della finestra e il caricamento dell'immagine. Il metodo main() non farà altro che istanziare un oggetto di tipo NoFilter:

```
/**
 * NoFilter
 */
package it.edmaster.ioprogrammo.filtrigrafici;

import java.awt.image.BufferedImage;
import java.io.IOException;

public class NoFilter extends ImageFilterTestBed {

      public NoFilter() throws IOException {
            super();
      }

      BufferedImage filter(BufferedImage image)
                                                  {
            return image;
      }
      String getFilterName() {
            return "Neutro";
      }

      /**
       * @param args
       * @throws IOException
       */

      public static void main(String[] args) throws
                                    IOException {
            new NoFilter();
      }
}
```

Il risultato sarà quello visibile in *Figura 1*. Come si vede, l'immagine è riquadrata da un bordo bianco, all'esterno del quale è presente il filo nelle due tonalità di grigio. Questo è ulteriormente spaziato dal bordo della fine-

FILTRI BASE

La libreria di filtri di JH Labs dispone anche di filtri di base, che replicano e integrano quelli presenti nella piattaforma Java. Questi includono il blur, inversione di colore e altri. Diversamente da quelli predefiniti, i filtri JH Labs offrono più parametri di configurazione.

Fig. 1. *Visualizzazione dell'immagine senza manipolazioni*

stra. Si noti che non è possibile ridimensionare la finestra, in quanto è stato impostato il flag resizable a false sulla JFrame. Questo accorgimento è dovuto al fatto che per effetto del Layout Manager utilizzato, il bordo bianco non sarebbe rimasto aderente all'immagine, ma al bordo della finestra. Si sarebbe dunque ottenuto un effetto sgradevole.
Si noti che per funzionare è necessario avere, nella root del CLASSPATH, l'immagine di prova IMG_5950.jpg.

INTEGRARE I FILTRI

A questo punto è necessario scaricare i filtri dal sito http://www.jhlabs.com/ip/filters/download.html. Il file zip che scarichermo contiene sia i sorgenti sia il file JAR da includere nel proprio progetto. Inoltre è presente un file di Ant per ricompilare i sorgenti e ricreare il file Filters.jar.
Il primo filtro che verrà provato è il filtro marmorizzatore, il cui effetto sull'immagine di prova è illustrato in Figura 2

Fig. 2. *Immagine sottoposta al filtro marmorizzato*

La classe che produce questa finestra si chiama Marmo. Il suo codice sorgente è il seguente:

```
package it.edmaster.ioprogrammo.filtrigrafici;
import java.awt.image.BufferedImage;
import java.io.IOException;
import com.jhlabs.image.MarbleFilter;
public class Marmo extends ImageFilterTestBed {
        public Marmo() throws IOException {
                super();
        }

        BufferedImage filter(BufferedImage image)
                                                        {
                MarbleFilter filter = new
                                        MarbleFilter();
                BufferedImage dest = filter
                .createCompatibleDestImage(image,
                                image.getColorModel());
                filter.filter(image, dest);
                return dest;
        }
        String getFilterName() {
                return "Marmorizza";
        }

        /**
         * @param args
         * @throws IOException
         */

        public static void main(String[] args) throws
                                        IOException {
                new Marmo();
        }
}
```

Come si vede, la struttura è simile a NoFilter. Gli elementi peculiari sono contenuti nel metodo filter() e getFilterName(). Il secondo si limita a ritornare la descrizione "Marmorizza". Il primo esegue questa volta la manipolazione dell'immagine.
Il filtro utilizzato è MarbleFilter, presente nel package com.jhlabs.image. I passaggi per l'utilizzo di questi filtri sono pochi e omogenei

In tutte le applicazioni destinate alla versione 2.0 è consigliabile utilizzare le nuove classi di insiemi generiche anziché le controparti non generiche meno recenti, quale *ArrayList*

APPLICAZIONI COMPLETE

Sul sito di JH Labs è disponibile una interessante applicazione, chiamata Java Image Editor (http://www.jhlabs.com/ie/index.html) che implementa un semplice programma di manipolazione immagini che si basa sui filtri descritti nell'articolo.
Il vantaggio di utilizzare l'applicazione Java Image Editor è che esistono pannelli di configurazione per ciascun filtro, che permettono di variare tutti i parametri di un dato filtro visualizzandone un'anteprima.

per le diverse tipologie di filtro. Questi sono:

1. Istanziare il filtro creando un oggetto del tipo voluto. In questo caso il filtro è MarbleFilter.
2. Ottenere un'immagine di output per le operazioni di filtro. La si può produrre utilizzando il metodo createCompatible DestImage() presenti sull'oggetto filtro. A questo metodo va passata un'immagine di riferimento e un modello di colore. Il primo parametro non è altro che il parametro BufferedImage passato al metodo filter() da noi creato. Il modello di colore può essere invece ottenuto utilizzando il metodo getColorModel() presente nella classe BufferedImage.
3. Impostare eventuali opzioni sul filtro (in questo esempio non ne sono state impostate).
4. Chiamare il metodo filter() sull'oggetto filtro, passando l'immagine di partenza e quella di destinazione prodotta nel punto 2.

L'immagine di destinazione viene poi riportata come risultato dal metodo.

GIOCARE CON I FILTRI

Ora che abbiamo visto come operare con un semplice filtro, vediamo altri esempi, anche di complessità crescente. Il prossimo filtro analizzato riproduce l'aspetto di un dipinto ad olio. Cambia i colori dell'immagine, sfocando i margini come si può vedere in Figura 2.

Fig. 3. *Immagine sottoposta al filtro olio*

Il filtro in questo caso è OilFilter e il codice sorgente è similare al precedente:

```
package it.edmaster.ioprogrammo.filtrigrafici;
import java.awt.image.BufferedImage;
import java.io.IOException;
import com.jhlabs.image.OilFilter;
public class Olio extends ImageFilterTestBed {
        public Olio() throws IOException {
                super();
        }
        BufferedImage filter(BufferedImage image){
                OilFilter filter = new OilFilter();
                BufferedImage dest =
        filter.createCompatibleDestImage(image,
                        image.getColorModel());
                filter.filter(image, dest);
                return dest;
        }
        String getFilterName() {
                return "Olio";
        }

        /**
         * @param args
         * @throws IOException
         */

        public static void main(String[] args) throws
                                IOException {
                new Olio();
        }
}
```

GESTIONE DEI PARAMETRI

Sebbene non siano stati utilizzati negli esempi precedenti, tutti i filtri dispongono di metodi per impostare o recuperare i parametri di funzionamento. Data la quantità di parametri disponibili è possibile una grande variabilità di risultati. In alcuni casi il risultato di un filtro, portato ai suoi limiti, produce un risultato simile a un altro filtro utilizzato con i parametri di default.

Il prossimo esempio utilizza il filtro CrystallizeFilter, che produce un effetto di cristallizzazione dell'immagine. L'output dell'esempio è illustrato in Figura 4.

La manipolazione non utilizza i valori di default, in quanto nel codice è presente l'impostazione di alcune opzioni. Il codice sorgente della classe Cristallizza, che impiega questo filtro, è il seguente:

ESTENDERE JAVA IMAGE EDITOR

È possibile scrivere plug-in per Java Image Editor, utilizzando la documentazione descritta alla pagina http://www.jhlabs.com/ie/plugin.html. Può essere una sfida interessante, che permetterebbe di sfruttare un'applicazione esistente e funzionante per utilizzare il proprio filtro o componente di trattazione delle immagini.

Fig. 4. **Immagine sottoposta al filtro cristallizzazione**

```
package it.edmaster.ioprogrammo.filtrigrafici;
import java.awt.image.BufferedImage;
import java.io.IOException;
import com.jhlabs.image.CrystallizeFilter;
public class Cristallizza extends ImageFilterTestBed {
        public Cristallizza() throws IOException {
                super();
        }

        BufferedImage filter(BufferedImage image){
                CrystallizeFilter filter = new
                                        CrystallizeFilter();
                BufferedImage dest =
filter.createCompatibleDestImage(image,
                                image.getColorModel());
filter.setGridType(CrystallizeFilter.OCTAGONAL |
                                CrystallizeFilter.SQUARE);
                filter.setScale(15.8f);
                filter.filter(image, dest);
                return dest;
        }

        String getFilterName() {
                return "Cristallizza";
        }

        /**
         * @param args
         * @throws IOException
         */

        public static void main(String[] args) throws
                                        IOException {
                new Cristallizza();
        }
}
```

Come si vede la struttura della classe è identica agli esempi precedenti, ma questa volta sono state impostate due opzioni. La prima è il tipo di griglia, che è il risultato dell'unione delle due costanti OCTAGONAL e SQUARE. Questo indica al filtro di utilizzare una griglia composta da ottagoni e quadrati. Ovviamente si sarebbe potuto passare solo OCTAGONAL o SQUARE, oppure una delle altre costanti definite nella classe. Queste costanti contengono dei valori che è possibile combinare in modo binario. In questo caso è stata compiuta un'unione con l'operatore |. L'altro parametro impostato è il valore di scala, che indica la dimensione di ciascun tassello che compone il risultato finale. Si noti che questo filtro offre anche altre opzioni, che non sono state utilizzate.

CONCLUSIONI

La libreria di filtri di JH Labs è molto interessante e completa.
Qui sono stati provati quattro filtri, appartenenti alla tipologia di quelli che cambiano lo stile dell'immagine. Ma questa libreria offre molto di più, a partire dai filtri blur e di correzione del colore. La prima categoria individua filtri di sfocatura e racchiude circa una dozzina di filtri. Un esempio di Motion Blur è illustrato in Figura 5.

Fig. 5. **Immagine sottoposta al filtro Motion Blur**

La seconda categoria permette la manipolazione dei colori dell'immagine.
Anche in questo caso i filtri disponibili sono molteplici, più di una dozzina. Oltre a questi sono disponibili filtri che implementano Texture, che producono effetti (cromo, specchio, feedback e glint) e per la distorsione delle immagini. In questo ultimo gruppo sono presenti filtri per introdurre bordi, distorsioni a forma di sfera, a diffusione, a cerchio, eccetera.
I filtri presenti sono moltissimi, quindi non resta altro che... sperimentare!

Massimiliano Bigatti

SOFTWARE SUL CD

AXIS1.4

IL FRAMEWORK PER CREARE WEB SERVICES IN JAVA

I Web Services rappresentano il futuro della programmazione. Sostanzialmente si tratta di miniapplicazioni che girano allo stessa stregua di un'applicazione web ma che esportano interfacce consumabili da client. Il linguaggio comune fra client è web services è SOAP, ovvero uno standard un protocollo standard su cui si basa lo scambio dei messaggi fra le parti. Ma come si crea un web services? Per quanto riguarda Java ci viene incontro questo interessante framework, prodotto, come spesso accade, dall'insostituibile Apache Foundation. Axis viene installato come un'application di Tomcat e a questo punto mette a disposizione interessanti metodi per la creazione facilitata dei web services e dei loro consumer

Directory:/Axis14

CAYENNE 1.2

IL NUOVO OBJECT RELATIONAL MAP

Noi programmatori siamo abituati a pensare ad oggetti tranne quando abbiamo a che fare con SQL. Il linguaggio standard dei database è abituato a lavorare per righe, celle e colonne, concetti che poco hanno a che fare con i nostri comodissimi oggetti e classi. Per ovviare a questo inconveniente sono nati gli Orbs. Tool intermedi che convertono dinamicamente tabelle, colonne, righe e celle, in classi ed oggetti. Cayenne è un tool emergente che appartiene a questa categoria e di cui ci parla Ivan Venuti in questo stesso numero di ioProgrammo.

Directory:/ Cayenne

ECLIPSE SDK 3.1.2

L'AMBIENTE APERTO

Eclipse è un IDE "Aperto" multipiattaforma e completamente scritto in Java. Per aperto si intende un sistema che può essere facilmente espandibile per mezzo di moduli. Perciò se da un lato inizialmente si presenta pronto per favorire lo sviluppo di applicazioni Java, dall'altro lato tramite moduli può diventare un comodo IDE per PHP oppure per C++ oppure per XML, oppure per qualunque altro linguaggio che sia supportato da un modulo. In questo numero di ioProgrammo lo abbiamo utilizzato in più occasioni ed è ormai diventato uno standard per ogni programmatore Java.

Directory:/ EclipseSDK312

ITEXT SHARP 3.1.1

IL PDF È FATTO

iText è una libreria OpenSource scritta in Java che consente di dotare le nostre applicazioni di funzionalità di esportazione verso il formato PDF. Si tratta di una libreria decisamente importante. Il PDF è un formato universale che garantisce una grande precisione nella stampa e una portabilità senza precedenti. Il poter creare documenti in formato PDF da una nostra applicazione sicuramente ne innalza il valore. D'altra parte itext non solo gestisce la mera esportazione dei dati in formato PDF ma dispone di funzionalità specifiche per crittografare o firmare digitalmente i documenti così creati. In questo numero vi presentiamo il porting della libreria verso .NET per poterla utilizzare in applicazioni C#.

Directory:/ iTextSharp

JAVA DEVELOPMENT KIT 1.5.0 UPGRADE 7

INDISPENSABILE PER PROGRAMMARE IN JAVA.

Ci corre l'obbligo, prima di tutto, di fare i nostri migliori auguri a java che ha appena compiuto 10 anni di vita. Il linguaggio di Sun nato con l'ambizione di essere il primo realmente multipiattaforma e che portava con se la grande ambizione di servire qualunque tipo di periferica, dopo 10 anni di esistenza può dire di avere largamente realizzato i suoi obiettivi. Con una base di programmatori larghissima e con il primato invidiabile di essere il linguaggio base per i telefonini di nuova generazione come per i PC come per i sistemi embedded è ad oggi uno dei linguaggi più diffusi al mondo. Per programmare in Java è necessario semplicemente dotarsi del J2SE che vi presentiamo in questo stesso numero

Directory:/ J2SE50Update7

JCAPTCHA 1.0RC3

REGISTRAZIONI SICURE SUI SITI WEB

State per registrarvi su un sito web, e assieme alle informazioni di registrazione vi tocca ricopiare nella form anche un codice riprodotto nel sito su una jpeg. Quello è un Captcha. In questo numero di ioProgrammo vi proponiamo un articolo di Ivan Venuti che mostra come creare un web services che consente a qualunque sito di implementare questa funzionalità. L'articolo è basato appunto su JCaptcha, questa interessante, quanto leggera libreria OpenSource scritta in Java che mette a disposizione appunto funzionalità per la creazione di Captcha.

Directory:/ JCaptcha

MYSQL MIGRATION KIT 1.0.25

MIGRA I DATI A MYSQL IN UN CLIC

MySQL si sta sempre affermando di più come uno standard de facto nel campo dei database. Sempre più frequente si manifesta la volontà di passare le proprie applicazioni dall'uso di un particolare DB a MySQL. Ad esempio è il caso di Access, SQL Server, Oracle, ciascuno dei quali in particolari condizioni si rivela inadeguato per un certo tipo di applicazione. Da oggi la migrazione è semplificata con questo

MySQL Migration Kit che con una procedura semplificata consente di passare facilmente da un formato di database ad un altro.

Directory:/ MySQLMigration1025

MYSQL ADMINISTRATOR

L CERVELLO DI MYSQL

Un buon database non si basa solo sulle performance o sulle funzionalità , ma deve anche essere semplice da amministrare, la dove per amministrazione si intende la creazione di nuovi utenti, la gestione dei permessi sulle tabelle, la creazione delle repliche e il tuning del sistema. Tutto questo si traduce in complessità di amministrazione. MySQL Administrator è una comodissima interfaccia grafica che vi consente di eseguire praticamente tutte le operazioni di gestione del database, dalla creazione al backup, alla gestione delle statistiche e dei log, svincolandovi da tutte quelle che sono le limitazioni di un'interfaccia a linea di comando.

Directory:/ MySQLAdmnistrator

MYSQL 5.0.21

IL DB PIÙ USATO SUL WEB

In congiunzione a PHP, MySQL fa girare buona parte del Web. Non c'è applicazione PHP che non utilizzi in qualche modo MySQL per gestire e recuperare i dati. Una così enorme diffusione si può giustificare solo con la bontà del prodotto. Infatti MySQL è un database estremamente leggero, gratuito, facile da usare e configurare. Si certo, manca ancora di qualcuna delle raffinatezze che fanno degli altri database dei mostri sacri per la programmazione in ambiente business, tuttavia anche questo gap va lentamente colmandosi. Per questi motivi MySQL sta rapidamente scalando posizioni anche nella programmazione di applicazioni standalone, anche se il suo ambito di riferimento primario rimane il web. In questo numero di ioProgrammo vi dedichiamo un megaarticolo che mostra come ottimizzare alcuni aspetti di MySQL per rendere le vostre applicazioni estremamente veloci.

Directory:/ MySQL5021

MYSQL QUERY BROWSER 1.1.20

IL TOOL PER NAVIGARE FRA GLI OGGETTI DI MYSQL

Come eseguire query senza dover interagire con la scomoda linea di comando di MySQL? Allo stesso modo, come navigare fra le tabelle di MySQL, cercare dati etc? ecco che ci viene incontro MySQL Query Browser, una potente interfaccia grafica che ci mette a disposizione numerosi strumenti per la gestione dei dati. Si tratta di un tool piuttosto interessante, anche se non ha la complessità di phpMyAdmin, tuttavia la sua natura di applicazione Standalone lascia intravedere ampi margini di miglioramento

NICKEL AND DIME CONTENT MANAGEMENT SYSTEM

UN CONTENT MANAGEMENT SYSTEM SCRITTO IN ASP.NET

Interessante questo CMS che implementa una serie di caratteristiche avanzate, come una granulare gestione degli utenti, un editor WYSIWIG un template manager, un search engine e ancora molto altro. Il cms usa un'interfaccia intermedia che gli consente di utilizzare qualunque database in modo del tutto trasparente all'utente. In questo modo è possibile utilizzarlo su hosting che mettono a disposizione la piattaforma Microsoft ma non SQL Server, ovvero la stragrande maggioranza

Directory:/ ndcms

Ruby on rails

IL FRAMEWORK VINCITORE DEL PREMIO "BEST TOOL" DEL 2006

Chi lo avrebbe mai detto che un framework basato su Ruby avrebbe vinto un premio così prestigioso? Ed invece ad anni di distanza dalla presentazione di Ruby, ecco arrivare un premio che lo consacra fra i linguaggi più interessanti della rete. Ruby On Rail è un framework per lo sviluppo di applicazioni Web, solide e affidabili. Soprattutto la bassa curva di apprendimento di Ruby e l'ottima progettazione del framework ne fanno un'accoppiata molto interessante

NANT 0.85

L'ASSEMBLATORE DI PROGETTI

Nant è un tool che consente di costruire un processo automatico per la generazione di applicazioni. Da utilizzare quando un prodotto è suddiviso in molte dll o sottoprodotti gestiti da reparti separati, nant crea una specie di processo batch che raccoglie i file dalle directory dei singoli sviluppatori, li compila e li predispone per la costruzione di un'applicazione completa. Si tratta dell'omologo di Ant per Java e risulta molto comodo in progetti di grandi dimensioni, oppure quando si fa riferimento a una serie di DLL che interagiscono fra loro.

Directory:/ Nant

POSTGRES SQL 8.1.4

COMPLETO E OPENSOURCE

Fra i database OpenSource non ne abbiamo trovato ancora uno in grado di competere per quantità di funzioni implementate! e a dire il vero pochi sistemi commerciali reggono il confronto con PostgreSQL. Al di là della velocità e dell'affidabilità del prodotto sono le sue caratteristiche ad averci impressionato. Foreign Keys, stored procedure, view, ereditarietà delle tabelle sono solo alcune delle caratteristiche che fanno di PostgreSQL un database completo. Particolarmente interessante è l'estendibilità del prodotto a mezzo di moduli esterni. Ne esistono già tantissimi che implementano praticamente tutto lo scibile sui database. Le caratteristiche interne al db più quelle aggiunte da moduli esterni ne fanno un prodotto completo e fuori dal comune.

Directory:/ PostgreSQL

PGADMIN 3.1.4.2

L'INTERFACCIA DI GESTIONE DI POSTGRES

Che PostgreSQL sia un database eccezionalmente potente è ormai noto. Purtroppo a tanta potenza corrisponde naturalmente una certa complessità nell'amministrazione delle tante caratteristiche che il prodotto espone. Per non perderci all'interno delle varie opzioni e delle possibilità di configurazione di PostGreSQL ecco che ci viene incontro PGAdmin, una comoda interfaccia di gestione che consente di amministrare l'intero database e tutte le sue caratteristiche più avanzate in modo visuale, senza

dover ricorrere alla scomoda linea di comando
Directory:/ PGAdmin

PHP 5.1.4
IL LINGUAGGIO DI INTERNET

Si tratta del linguaggio con il quale sono sviluppate la maggior parte delle applicazioni internet esistenti. La curva di apprendimento è bassissima, le funzionalità esposte elevatissime, certamente se avete intenzione di sviluppare per il web non potrete fare a meno di provare anche questo linguaggio come base per le vostre applicazione. Attualmente la versione 5.1.4 espone un modello ad oggetti piuttosto completo che rende PHP un linguaggio moderno e persino didattico per chi vuole imparare a programmare ad oggetti senza dover affrontare la complessità di Java o .NET
Directory PHP 5.1.4

PYTHON 2.4.3
IL LINGUAGGIO EMERGENTE

Un linguaggio di scripting, multipiattaforma, orientato agli oggetti. Tre caratteristiche che rendono questo linguaggio piuttosto appetibile. Non per niente tutte le classifiche mondiali sulla diffusione dei linguaggi di programmazione lo individuano come quello maggiormente in ascesa. Python è elegante, estensibile, con una curva di apprendimento non elevatissima. Viene utilizzato per programmare moltissimi tool di gestione dei sistemi Unix, ma sta trovando una rapida applicazione anche sui sistemi windows e nello sviluppo di applicazioni crossplatform. Python recentemente sta diventando davvero un punto di riferimento fisso per moltissimi sviluppatori.
Directory:/Python

SPRING 1.2.8
IL FRAMEWORK CHE LI RACCHIUDE TUTTI

In questo numero lo utilizziamo per costruire un guestbook piuttosto particolare. Nell'articolo del bravo Ivan Venuti si mostra come implementare il pattern MVC con Spring. Si tratta di un framework che sta ottenendo un grande successo, per due motivi fondamentali. Prima di tutto raccoglie sotto un unico cappello una serie di tecnologie già esistenti, donandogli però una certa interoperabilità e una forma omogenea. Secondo, implementa perfettamente il pattern Inversion Of Control che consente un alto grado di disaccoppiamento tra le classi di modo che la modifica di una non cambi il comportamento di molte altre. Si tratta di un pattern molto interessante, per quanto sia investito di una certa complessità.
Directory:/Spring128

MYSQL CONNECTOR
IL PONTE VERSO I LINGUAGGI

Un database da solo non sarebbe niente se non fosse possibile sviluppare applicazioni che consentano di immagazzinare e manipolare i dati.
Le applicazioni sono fatte dai linguaggi di programmazione. Per cui deve esistere qualcosa per collegare i linguaggi al database, nel caso di MySQL questo software "ponte" prende il nome di connectors, in questo numero presentiamo quello per Java e quello per .NET, per PHP il connectors è già integrato nel linguaggio, per tutti gli altri c'è ODBC il driver universale per i database.
Directory:/ MySQLConnector

Ruby 1.8.2
IL NUOVO CHE AVANZA

E' così quest'anno il premio come miglior prodotto dell'anno va a RubyOnRail un framework basato su Ruby. Quando qualche anno fa questo linguaggio fece la sua comparsa sul mercato, nessuno gli diede molto credito. Ad oggi si presenta come un linguaggio di scripting eccezionale che condensa in un'unica soluzione un modello ad oggetti completo, una curva di apprendimento che rasenta quasi lo zero assoluto, un alta integrazione con gli oggetti com di windows, e altre elementi ancora.

TOMCAT 5.5.17
L'APPLICATION SERVER PER JSP

Se siete dei programmatori JSP avete senza dubbio bisogno di Tomcat per testare e utilizzare le vostre applicazioni. E' senza dubbio un server web ovvero può funzionare da server web ma è soprattutto un Application Server che vi consente di utilizzare la tecnica delle servlet o delle JSP per creare applicazioni Web con una logica business e con il linguaggio Java a fare da asse portante. Tomcat ha in se anche un Web Server, ma quel che più conta è un container di applicazioni Java.
Directory:/Tomcat

RAD RAILS 0.6.3
L'AMBIENTE PER LO SVILUPPO DI APPLICAZIONI BASATE SU RUBYONRAILS

Utile e ben fatto questo ambiente RAD interamente basata su Ruby On Rails, va a completare un trittico dalle incredibili potenzialità per chi vuole sperimentare linguaggi alternativi ma molto ben strutturati per lo sviluppo di web application
Directory:/ RadRails

YETANOTHERFORUM 1.0.1
ANCORA UN ALTRO FORUM!

Il gioco di parole si presta bene per marcare la vocazione OpenSource di questo progetto. Di fatto, tipicamente questo genere di gioco di parole viene applicato proprio a software OpenSource. YetAnotherForum è un forum scritto interamente in .NET. Il progetto è piuttosto ambizioso e si pone sulla scia di progetti ben più noti quali phpBB. Tuttavia è importante notare oltre alle qualità intrinseche di questo forum che sono veramente eccellenti, quanto anche in ambienti .NET si stia cominciando a diffondere il concetto di OpenSource.
Directory:/yetanotherforum

TBLOGGER 0.23
UN BLOG INTERAMENTE SCRITTO IN .NET

Interessante questo piccolo sistema di blog scritto son il .NET framework. Certo non si può dire che sia un sistema completo, anzi a dire il vero si tratta di un blog molto semplice, ma assolve a due funzionalità. La prima interamente didattica, di fatto la semplicità del codice si pone come un ottima base per comprendere come si possano sviluppare sistemi di questo genere. La seconda ovviamente in sistemi di produzione la dove non si necessitino di funzionalità troppo elevate.
Directory:/ Tblogger

ALBERI: IMPARIAMO COME BILANCIARLI

QUAL È IL MODO MIGLIORE PER ORDINARE I DATI? E COSA SUCCEDE QUANDO UN DATO ENTRA O ESCE DA UNA STRUTTURA ORDINATA? A QUESTA E A MOLTE ALTRE DOMANDE DAREMO UNA RISPOSTA IN QUESTO ARTICOLO

Abbiamo avuto modo di visionare gli alberi binari e quelli di ricerca, che molto sinteticamente, per chi ha saltato lo scorso numero, rivisiteremo. Ci occuperemo adesso dei particolari alberi che godono di una significativa caratteristica che li rende "equilibrati ed armoniosi" in un'unico termine: "bilanciati". Metteremo le mani nella parte meramente tecnica di manipolazione della struttura, comprendendo al meglio gli strumenti e i metodi da usare nelle diverse occasioni. Vedremo che adottare alcune tecniche di ottimizzazione consente di effettuare scansioni realmente rapidissime. Analizzeremo anche alcuni casi estremi in cui la presenza di dati particolarmente sbilanciati non consente di realizzare strutture perfette. Ad esempio cosa succede se volete ordinare l'alfabeto? si degenera nel caso in cui un albero diventa una lista e i metodi che esploreremo non sono più applicabili.

ALBERI BINARI E DI RICERCA

Un albero è una struttura dati molto flessibile che ci consente di rappresentare molti problemi. Con albero nell'ambiente dei programmatori si intende di fatto un albero binario. Una struttura cioè che abbia per ogni elemento (nodo) al più due discendenti. In figura 1 è mostrato un semplice albero binario.

Fig. 1: ***Albero binario di ricerca***

Un albero è di ricerca se come nella figura i dati sono sistemati in modo da essere ordinati. Per ogni nodo tutte le informazioni contenute nel sottoalbero sinistro sono minori e quelle del sottoalbero destro sono maggiori. La volta scorsa abbiamo approntato la strutturazione fisica e le principali funzioni di visita, ricerca e inserimento. Il tutto seguendo le indicazioni della OOP (Object oriented programming), ossia la programmazione ad oggetti. In C++ è stata costruita una classe albero. Ne riporto il prototipo arricchito dei nuovi metodi che prevedono la gestione degli alberi avl.Nell'articolo pubblicato nel numero 103 di ioProgrammo abbiamo appunto analizzato nel dettaglio tutti i metodi esposti in questo prototipo, eccetto il caso della rimozione che vedremo in questo numero

```
class albero

{  public: albero();
         albero(char*);
         ~albero();
      void aggiungi(char*);
      void postordine();
      void preordine();
      void inordine();
      bool ricerca(char*);
      void rimuovi(char*);
      void iterat();
      int calcolaprof();
      bool bilanciatoavl();
   private:
   struct nodo {
      char *info;
      nodo *sx;
      nodo *dx;
    };
    struct nodo *tree;
    int prof; // profondità calcolata con il metodo
                                              associato;
    nodo* q;
```

```
  nodo* p;
  void agg(nodo* &, char*);
  void postordinepriv(nodo*);
  void preordinepriv(nodo*);
  void inordinepriv(nodo*);
  void cerca(nodo*, char*, bool &);
  void rim(nodo* &, nodo*);
  void elim(nodo* &, char*);
  bool bilanciatoavlpriv(nodo *);
  int profondita(nodo*);
  int maggiore(int,int);
};
```

Tra i membri pubblici si scorgono le varie funzioni per effettuare la vista e l'inserimento. Nei prossimi paragrafi analizzeremo in modo particolareggiato i nuovi componenti.

RIMOZIONE DI UN NODO

Cominciamo lo studio di un caso oltre che utile, molto istruttivo. La rimozione di un nodo da un albero. In altri termini si vuole cancellare un nodo che abbia contenuto informativo uguale a quello che specifichiamo come utenti da input, e naturalmente lasciare la struttura a forma di albero. Va detto che non è un'operazione banale come l'inserimento di un elemento. Il compito è semplice se c'è da rimuovere una foglia. In tal caso è sufficiente cancellarla. Anche la cancellazione di un nodo che abbia un solo discendente (o destro o sinistro, non importa quale) non è difficile. Basta "attaccare" il sottoalbero discendente che è unico al posto dell'elemento da eliminare. I problemi sorgono quando si deve cancellare un qualsiasi altro nodo diverso da quelli prima descritti. Insomma, quando un nodo ha due discendenti. Si comprende che è impossibile collegare i due elementi ad un solo puntatore. Bisogna quindi procedere sostituendo l'elemento rimosso con il nodo più a destra del sottoalbero sinistro o con il nodo più a sinistra del sottoalbero destro, entrambi i quali hanno al più un discendente. In figura 2 sono riportati i tre casi di rimozione.

Fig. 2: ***Alcuni esempi di cancellazione di nodi da un albero. A albero iniziale. B cancellazione di una foglia. C cancellazione di un nodo con un solo discendente. D cancellazione di un nodo intermedio***

Nell'implementare il procedimento si è costruito un metodo pubblico rimuovi(char* mess), a tale proposito si esamini la classe completa di tutti i suoi membri. Tale metodo usa altre due funzioni private che attraverso chiamate ricorsive implementano la linea di azione prima descritta. Vediamo il codice.

```
// routine per l'implementazione di rimuovi

void albero::rim(nodo* &nod, nodo* nod2)
{

  if (nod->dx!=0) rim(nod->dx,nod);
  else
  { nod2->info=nod->info;
       q=nod; // salvataggio per delete()
       nod=nod->sx;
       };
};

// routine per l'implementazione di rimuovi

void albero::elim(nodo* &p,char *mess)
{  if (p==0) cout<<"parola non presente\n";
    else if (strcmp(mess,p->info)<0) elim
                                  (p->sx,mess);
         else if (strcmp(mess,p->info)>0) elim
                                  (p->dx,mess);
             else // si esaminano i tre casi
             {  q=p;  // salvataggio per delete()
                if (p->dx==0) p=p->sx;
                else if (p->sx==0) p=p->dx;
                    else rim(p->sx,p);
             };
};

// metodo pubblico per la rimozione del nodo
                                   con info mess
void albero::rimuovi(char* mess)
{
  q=0;
  p=tree;
  elim(p,mess);
  if (q!=0) delete q;
};
```

In q ci sarà il nodo da eliminare fisicamente mediante una delete(), che libera opportunamente la memoria. Mentre p è il puntatore che scandaglia la struttura ed è inizialmente posizionato sulla radice dell'albero (tree). Con elim si pri-

ma verifica se c'è il messaggio come contenuto informativo di un nodo. Fin quando non si trova si percorre l'albero come per una normale ricerca. Una volta trovato l'elemento, se uno dei due discendenti è nullo (non a caso oltre che lo zero si può usare la parola chiave NULL) si attacca il sottoalbero non nullo. Non si controlla che entrambi i discendenti siano nulli, situazione che si presenta per una foglia; visto che l'azione suddetta funziona anche in questo caso; semplicemente si attacca non un sottoalbero ma un puntatore nullo. L'ultimo caso si risolve ricorrendo alla funzione rim. Essa scende nel ramo più a destra del sottoalbero sinistro e individua il nodo da sostituire. Si assegna il contenuto informativo. Al termine delle operazioni ci si libera con delete() del nodo foglia raggiunto.

Fig. 3: ***Esempio di albero binario di ricerca non bilanciato***

ALBERI AVL

Normalmente la struttura albero non rimane invariata durante l'applicazione. Spesso subisce significative modifiche, usualmente cresce. È il caso del problema delle concordanze dove un nodo contiene una singola parola e la sua occorrenza in un testo; quindi due campi informativi. In tal senso nell'analizzare il testo ogni qual volta si incontra una nuova parola si aggiunge un nuovo nodo e si inizializza a uno(1) il numero di occorrenze. Se si incontra una parola non nuova semplicemente se ne incrementano le occorrenze. Strutture siffatte sono molto comuni nell'ambito della linguistica computazionale. Si tratta di un esempio significativo perché l'albero cresce in modo rapido; è quindi importante che sia efficiente. Sarebbero non ottimizzate strutture come quella mostrata in figura 3.

In realtà se non si adottano misure non si può prevedere come l'albero crescerà. Vi sono dei casi limite, che ovviamente hanno bassissime probabilità di verificarsi per i quali un albero degenera in una lista. Se, infatti tutte le parole che in sequenza si analizzano sono strettamente crescenti oppure decrescenti vengono appese sempre a destra oppure a sinistra. Cosicché, la ricerca degenera a complessità n/2 e non log(n) come si ci auspicava. Il caso migliore è che l'albero sia perfettamente bilanciato. Un albero è perfettamente bilanciato se per ogni nodo il numero di nodi del sottoalbero sinistro e del sottoalbero destro differiscono al più di 1.

Si tratta di una situazione ideale che per essere raggiunta e mantenuta richiede un notevole dispendio di tempo. Nella stragrande maggioranza dei casi non conviene mantenere un equilibrio così rigido della struttura. Per superare un bilanciamento così rigido è stato introdotto una definizione per cosi dire più flessibile che è anche più facile da raggiungere e mantenere. Un albero è bilanciato AVL se per ogni nodo le profondità dei suoi due sottoalberi differiscono al più di uno. Quindi ci si concentra sulla profondità anziché sul numero di nodi. Tale definizione è stata formulata dagli studiosi G.M. Adelson Velskii e E.M. Landis, le cui iniziali hanno dato il nome agli alberi. Con il termine alberi bilanciati si intende di fatto gli alberi bilanciati AVL. Gli alberi bilanciati sono un sottoinsieme degli alberi perfettamente bilanciati. Come vedremo gli AVL oltre che essere distinti da una semplice definizione sono associati a semplici routine per mantenerne il bilanciamento. Inoltre, è sempre garantita complessità logaritmica, anche nel caso peggiore, per operazioni di ricerca, inserimento e cancellazione. Analizziamo qualche altro metodo associato agli alberi AVL. Vediamo come si può verificare la caratteristica di bilanciamento.

```
// routine per l'implementazione di profondità

int albero::maggiore(int x, int y)
{  int z;
   if (x>y) z=x;
```

CONVIENE BILANCIARE ... PERFETTAMENTE!

Il bilanciamento di un albero ha un costo, in termini di complessità computazionale, quindi di tempo. Tale costo è molto più elevato nel caso del perfetto bilanciamento. Un'analisi presentata da N. Wirth ha dimostrato che il costo che si deve sostenere per mantenere un albero perfettamente bilanciato, nella maggior parte dei casi non viene giustificato da una più efficiente utilizzo dell'albero stesso. Conviene perfettamente bilanciare solo nei casi limiti per i quali l'albero degenera in liste o in strutture "simili" a liste, che però sono appunto molto improbabili.

```
   else z=y;
   return z;
};

// metodo privato di profondita

int albero::profondita(nodo* nod)
{
   if (nod==0) prof=0;
   else prof=1+maggiore(profondita(nod->sx),
                        profondita(nod->dx));
   return prof;
};

// metodo pubblico per il calcolo della profondità

int albero::calcolaprof()
{  return profondita(tree);
};

// metodo privato per la verifica
// di bilanciamento avl

bool albero::bilanciatoavlpriv(nodo* nod)
{
  bool bilanc;
   if (nod==0) bilanc=true;
   else bilanc= ( (abs(profondita(nod->sx)
                 profondita(nod->dx))<=1)  &&
            (bilanciatoavlpriv(nod->sx)) &&
                 (bilanciatoavlpriv(nod->dx)) );
   return bilanc;
};

// metodo pubblico per la verifica
// di bilanciamento avl

bool albero::bilanciatoavl()
{  return bilanciatoavlpriv(tree);
};
```

ANALIZZATORE SINTATTICO DI ESPRESSIONI

Un problema molto interessante che trova la sua naturale soluzione con l'uso di un albero binario è l'analizzatore sintattico di espressioni matematiche. Data una stringa di input che contiene l'espressione si vuole generare in output il risultato. L'espressione può avere un numero qualsiasi di parentesi tonde. L'alfabeto che si considera, ossia l'insieme dei simboli che si possono trattare è: {0, 1, 2, 3,4 ,5 ,6, 7, 8, 9, *, /, +, -, <spazio>, $, (,), }. I simboli sono classificabili nei tre insiemi: delle cifre, delle operazioni e dei caratteri. L'ultimo insieme oltre che lo spazio e le parentesi contiene il simbolo $ di fine stringa. Ovviamente, si trattano solo numeri interi.

La struttura dell'algoritmo di cui forniremo le linee generali di azione può essere così schematizzata:

- leggi stringa
- analizza stringa
- trasforma stringa in albero
- valuta albero
- produci output

Come si può intuire si tratta di un piccolo compilatore costituito proprio da analizzatore lessicale (passo 2) e da parser (passo 4). Ovviamente, il tutto è semplificato, e non è necessario scomodare tutte i costrutti teorici dei compilatori. Il singolo nodo come deve essere fatto? Dovrà contenere il patrimonio informativo e un altro campo che specifica a quale insieme il simbolo appartiene, visto che per ogni insieme di simboli si dovrà attuare un diverso comportamento.

Vediamo come un'espressione genera un albero; la seguente di esempio: 4 + 6 – 1$ produce l'albero di figura 6.
Si nota ad esempio che un'operazione è sempre un nodo e che essa non può trovarsi come terminale (foglia) altrimenti risulterebbe incongruente il calcolo.

Fig. 6: albero binario associato ad un'espressione matematica

Inoltre, si nota come l'espressione si ottenga dall'albero con una visita simmetrica (in ordine).
L'analizzatore della stringa dovrà sostanzialmente verificare che simboli siano effettivamente appartenenti all'alfabeto e potrà ad esempio eliminare gli spazi che non servono per la costruzione. Anche la congruenza delle parentesi può essere fatta in questa fase. Nella trasformazione da stringa ad albero, che è un'operazione delicata si devono fare inserimenti mirati a seconda del tipo di simbolo che si incontra. Ad esempio, la semplice terna (cifra, operazione, cifra) si trasforma in un mini albero (fig 6 cerchiato in rosso)

Il metodo maggiore semplicemente individua il più grande tra due numeri, si tratta di una routine di servizio usata da profondita. Questo metodo (l'assenza dell'accento sulla a non è un errore, i programmatori sanno che à è un carattere non consentito per gli identificatori) calcola la profondità. Poiché prende come parametro un nodo, si è preferito renderlo privato

Fig. 4: Albero binario bilanciato su cui operare un inserimento

e produrre un altro metodo: calcolaprof() che sia pubblico. Nella definizione ricorsiva di profondità si realizza il metodo. La profondità è la somma tra 1 e la maggiore profondità tra il sottoalbero sinistro e il sottoalbero destro. Le profondità dei due sottoalberi vengono calcolate applicando l'identica definizione. Conoscendo la profondità si può facilmente verificare se l'al-

bero è bilanciato o meno. Anche in questo caso l'assegnazione alla variabile booleana bilanc, attraverso riferimenti ricorsivi alla stessa procedura, descrive in modo esplicativo il concetto di bilanciamento.

BIBLIOGRAFIA

N. Wirth - Algoritmi+ strutture dati uguale programmi – Tecniche nuove
G. Porcelli, L. Porcelli – Elementi di programmazione – Paravia
Kruse, Tondo, Leung – Data structured and program design in C – Prentice Hall international, Inc. Nel web: Raccomandato: http://webpages.ull.es/users/jriera/Docencia/AVL/AVL%20tree%20applet.htm

INSERIMENTO IN ALBERI BILANCIATI

Quando bisogna inserire un nuovo nodo in un albero bilanciato (intendiamo sempre AVL) si possono presentare tre casi rispetto alle profondità dei sottoalberi destro e sinistro della radice:

1. le profondità diventano diverse, ma il criterio di bilanciamento è salvaguardato,

2. le profondità diventano uguali. C'è un miglioramento della situazione

3. le profondità diventano diverse violando il criterio.

Risulta chiaro che bisogna prendere provvedimenti per ristabilire il bilanciamento solo nel caso 3. Rispetto all'albero di figura 4, provocano uno "sbilanciamento" gli inserimenti di nodi che verrebbero appesi al nodo 7, negli altri casi si mantiene il bilanciamento.
La condizione di equilibrio è che la differenza tra le profondità tra sottoalbero sinistro e destro della radice sia -1, 0 o 1. È fondamentale stabilire il modo di memorizzare tale informazione. Si può mantenere tale indice implicito alla struttura, quindi ricalcolarlo per ogni variazione della stessa.
Oppure, esplicito prevedendo spazi nei vari nodi per la memorizzazione delle profondità. Il primo metodo in caso di strutture molto estese richiede un costo computazionale elevato. Il secondo fornisce un maggiore spreco di memoria ma sicuramente migliora e non di poco le prestazioni.
Questa decisione si prende in fase di progettazione anche con riferimento al tipo di albero. Vediamo nello specifico come si ristabilisce il bilanciamento dopo un inserimento del tipo 3. I casi limite sono 2 e sono riproposti nella figura 5 a. Con la croce si indica il nuovo elemento aggiunto che non fa rispettare il criterio di bilanciamento. Si noti una differenza massima di profondità pari a 2.
Il bilanciamento si ottiene mediante operazioni di rotazione. La rotazione singola (caso 1), che sia destra o sinistra non fa differenza, effettua spostamenti verticali dei nodi, in orizzontale essi mantengono sempre la stessa posizione. Anche la doppia rotazione (caso 2) ristabilisce il bilanciamento con spostamenti verticali.

CONCLUSIONI

Capire i metodi elementari di manipolazioni di strutture come gli alberi bilanciati AVL è un importante patrimonio per il programmatore, che può spendere queste abilità per la risoluzione di un'ampia casistica di applicazioni che richiedono l'uso di alberi.
Per quanto gli strumenti di programmazione moderna, nascondano il più delle volte la complessità di questo genere di algoritmi, tuttavia nella programmazione tradizionale è importante riconoscere come certi metodi vengano applicati. In caso di necessità di ottimizzazione, essere in grado di scrivere un algoritmo di ricerca binaria su alberi bilanciati diventa spesso l'unica via d'uscita.

Fabio Grimaldi

Fig. 5: **Bilanciamento di alberi per rotazione**